Petri Nets for Systems and Synthetic Biology

Monika Heiner[1], David Gilbert[2], and Robin Donaldson[2]

[1] Department of Computer Science, Brandenburg University of Technology
Postbox 10 13 44, 03013 Cottbus, Germany
monika.heiner@tu-cottbus.de
[2] Bioinformatics Research Centre, University of Glasgow
Glasgow G12 8QQ, Scotland, UK
drg@brc.dcs.gla.ac.uk, radonald@brc.dcs.gla.ac.uk

Abstract. We give a description of a Petri net-based framework for modelling and analysing biochemical pathways, which unifies the qualitative, stochastic and continuous paradigms. Each perspective adds its contribution to the understanding of the system, thus the three approaches do not compete, but complement each other. We illustrate our approach by applying it to an extended model of the three stage cascade, which forms the core of the ERK signal transduction pathway. Consequently our focus is on transient behaviour analysis. We demonstrate how qualitative descriptions are abstractions over stochastic or continuous descriptions, and show that the stochastic and continuous models approximate each other. Although our framework is based on Petri nets, it can be applied more widely to other formalisms which are used to model and analyse biochemical networks.

1 Motivation

Biochemical reaction systems have by their very nature three distinctive characteristics. (1) They are inherently bipartite, i.e. they consist of two types of game players, the species and their interactions. (2) They are inherently concurrent, i.e. several interactions can usually happen independently and in parallel. (3) They are inherently stochastic, i.e. the timing behaviour of the interactions is governed by stochastic laws. So it seems to be a natural choice to model and analyse them with a formal method, which shares exactly these distinctive characteristics: stochastic Petri nets.

However, due to the computational efforts required to analyse stochastic models, two abstractions are more popular: qualitative models, abstracting away from any time dependencies, and continuous models, commonly used to approximate stochastic behaviour by a deterministic one. We describe an overall framework to unify these three paradigms, providing a family of related models with high analytical power.

The advantages of using Petri nets as a kind of umbrella formalism are seen in the following:

M. Bernardo, P. Degano, and G. Zavattaro (Eds.): SFM 2008, LNCS 5016, pp. 215–264, 2008.

- intuitive and executable modelling style,
- true concurrency (partial order) semantics, which may be lessened to interleaving semantics to simplify analyses,
- mathematically founded analysis techniques based on formal semantics,
- coverage of structural and behavioural properties as well as their relations,
- integration of qualitative and quantitative analysis techniques,
- reliable tool support.

This chapter can be considered as a tutorial in the step-wise modelling and analysis of larger biochemical networks as well as in the structured design of systems of ordinary differential equations (ODEs). The qualitative model is introduced as a supplementary intermediate step, at least from the viewpoint of the biochemist accustomed to quantitative modelling only, and serves mainly for model validation since this cannot be performed on the continuous level, and is generally much harder to do on the stochastic level. Having successfully validated the qualitative model, the quantitative models are derived from the qualitative one by assigning stochastic or deterministic rate functions to all reactions in the network. Thus the quantitative models preserve the structure of the qualitative one, and the stochastic Petri net describes a system of stochastic reaction rate equations (RREs), and the continuous Petri net is nothing else than a structured description of ODEs.

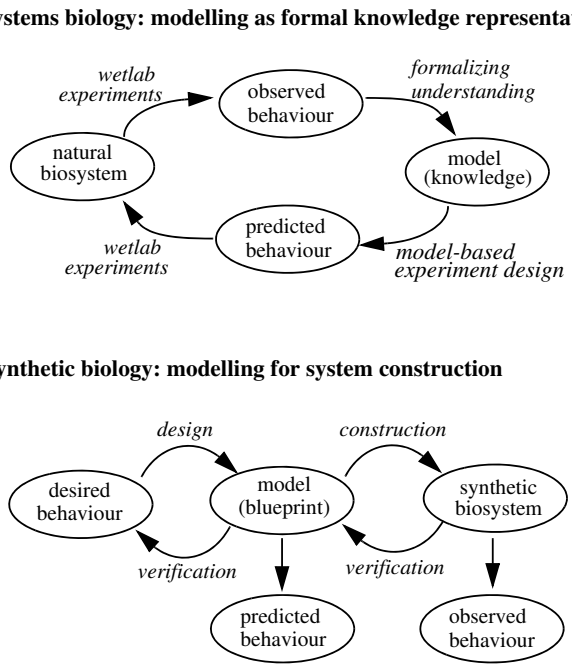


Fig. 1. The role of formal models in systems biology and synthetic biology.

This framework is equally helpful in the setting of systems biology as well as synthetic biology, see Figure 1. In systems biology, models help us in formalising our understanding of what has been created by natural evolution. So first of all, models serve as an unambiguous representation of the acquired knowledge and help to design new wetlab experiments to sharpen our comprehension.

In synthetic biology, models help us to make the engineering of biology easier and more reliable. Models serve as blueprints for novel synthetic biological systems. Their employment is highly recommended to guide the design and construction in order to ensure that the behaviour of the synthetic biological systems is reliable and robust under a variety of conditions.

Formal models open the door to mathematically founded analyses for model validation and verification. This paper demonstrates typical analysis techniques, with special emphasis on transient behaviour analysis. We show how to systematically derive and interpret the partial order run of the signal response behaviour, and how to employ model checking to investigate related properties in the qualitative, stochastic and continuous paradigms. All analysis techniques are introduced through a running example. To be self-contained, we give the formal definitions of the most relevant notions, which are Petri net specific.

This paper is organised as follows. In the following section we outline our framework, discussing the special contributions of the three individual analysis approaches, and examining their interrelations. Next we provide an overview of the biochemical context and introduce our running example. We then present the individual approaches and discuss mutually related properties in all three paradigms in the following order: we start off with the qualitative approach, which is conceptually the easiest, and does not rely on knowledge of kinetic information, but describes the network topology and presence of the species. We then demonstrate how the validated qualitative model can be transformed into the stochastic representation by addition of stochastic firing rate information. Next, the continuous model is derived from the qualitative or stochastic model by considering only deterministic firing rates. Suitable sets of initial conditions for all three models are constructed by qualitative analysis. Finally, we refer to related work, before concluding with a summary and outlook regarding further research directions.

2 Overview of the framework

In the following we describe our overall framework, illustrated in Figure 2, that relates the three major ways of modelling and analysing biochemical networks described in this paper: qualitative, stochastic and continuous.

The most abstract representation of a biochemical network is *qualitative* and is minimally described by its topology, usually as a bipartite directed graph with nodes representing biochemical entities or reactions, or in Petri net terminology *places* and *transitions* (see Figures 4 – 6). Arcs can be annotated with stoichiometric information, whereby the default stoichiometric value of 1 is usually omitted.

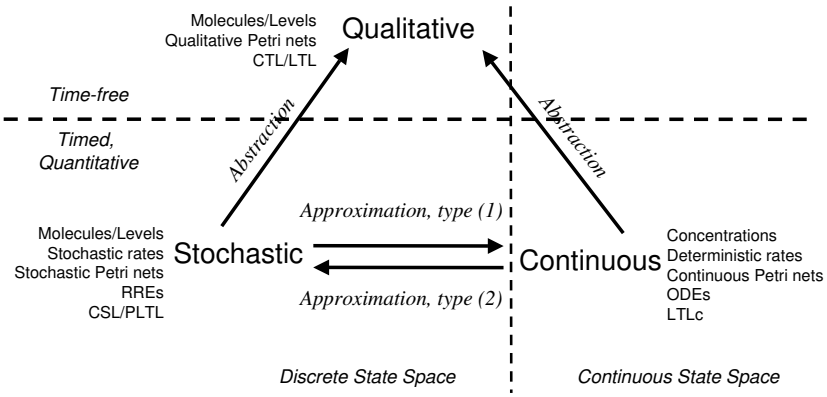


Fig. 2. Conceptual framework

The qualitative description can be further enhanced by the abstract representation of discrete quantities of species, achieved in Petri nets by the use of tokens at places. These can represent the number of molecules, or the level of concentration, of a species. The standard semantics for these qualitative Petri nets (QPN) does not associate a time with transitions or the sojourn of tokens at places, and thus these descriptions are time-free. The qualitative analysis considers however all possible behaviour of the system under any timing. The behaviour of such a net forms a discrete state space, which can be analysed in the bounded case, for example, by a branching time temporal logic, one instance of which is Computational Tree Logic (CTL), see [CGP01].

Timed information can be added to the qualitative description in two ways – stochastic and continuous. The stochastic Petri net (SPN) description preserves the discrete state description, but in addition associates a probabilistically distributed firing rate (waiting time) with each reaction. All reactions which occur in the QPN can still occur in the SPN, but their likelihood depends on the probability distribution of the associated firing rates. Special behavioural properties can be expressed using e.g. Continuous Stochastic Logic (CSL), see [PNK06], a probabilistic counterpart of CTL, or Probabilistic Linear-time Temporal Logic (PLTL), see [MC208], a probabilistic counterpart to LTL [Pnu81]. The QPN is an abstraction of the SPN, sharing the same state space and transition relation with the stochastic model, with the probabilistic information removed. All qualitative properties valid in the QPN are also valid in the SPN, and vice versa.

The continuous model replaces the discrete values of species with continuous values, and hence is not able to describe the behaviour of species at the level of individual molecules, but only the overall behaviour via concentrations. We can regard the discrete description of concentration levels as abstracting over the continuous description of concentrations. Timed information is introduced by the association of a particular deterministic rate information with each transition, permitting the continuous model to be represented as a set of ordinary differential

equations (ODEs). The concentration of a particular species in such a model will have the same value at each point of time for repeated experiments. The state space of such models is continuous and linear. So it has to be analysed by a linear time temporal logic (LTL), for example, Linear Temporal Logic with constraints (LTLc) in the manner of [CCRFS06], or PLTL [MC208].

The stochastic and continuous models are mutually related by approximation. The stochastic description can be used as the basis for deriving a continuous Petri net (CPN) model by approximating rate information. Specifically, the probabilistically distributed reaction firing in the SPN is replaced by a particular average firing rate over the continuous token flow of the CPN. This is achieved by approximation over hazard (propensity) functions of type (1), described in more detail in section 5.1. In turn, the stochastic model can be derived from the continuous model by approximation, reading the tokens as concentration levels, as introduced in [CVGO06]. Formally, this is achieved by a hazard function of type (2), see again section 5.1.

It is well-known that time assumptions generally impose constraints on behaviour. The qualitative and stochastic models consider all possible behaviours under any timing, whereas the continuous model is constrained by its inherent determinism to consider a subset. This may be too restrictive when modelling biochemical systems, which by their very nature exhibit variability in their behaviour.

3 Biochemical Context

We have chosen a model of the mitogen-activated protein kinase (MAPK) cascade published in [LBS00] as a running case study. This is the core of the ubiquitous ERK/MAPK pathway that can, for example, convey cell division and differentiation signals from the cell membrane to the nucleus. The model does not describe the receptor and the biochemical entities and actions immediately downstream from the receptor. Instead the description starts at the RasGTP complex which acts as a kinase to phosphorylate Raf, which phosphorylates MAPK/ERK Kinase (MEK), which in turn phosphorylates Extracellular signal Regulated Kinase (ERK). This cascade (RasGTP $\rightarrow$ Raf $\rightarrow$ MEK $\rightarrow$ ERK) of protein interactions controls cell differentiation, the effect being dependent upon the activity of ERK. We consider RasGTP as the input signal and ERKPP (activated ERK) as the output signal.

The scheme in Figure 3 describes the typical modular structure for such a signalling cascade, compare [CKS07]. Each layer corresponds to a distinct protein species. The protein Raf in the first layer is only singly phosphorylated. The proteins in the two other layers, MEK and ERK respectively, can be singly as well as doubly phosphorylated. In each layer, forward reactions are catalysed by kinases and reverse reactions by phosphatases (Phosphatase1, Phosphatase2, Phosphatase3). The kinases in the MEK and ERK layers are the phosphorylated forms of the proteins in the previous layer. Each phosphorylation/dephosphorylation step applies mass action kinetics according to the

following pattern: $A + E \rightleftharpoons AE \rightarrow B + E$, taking into account the mechanism by which the enzyme acts, namely by forming a complex with the substrate, modifying the substrate to form the product, and a disassociation occurring to release the product.

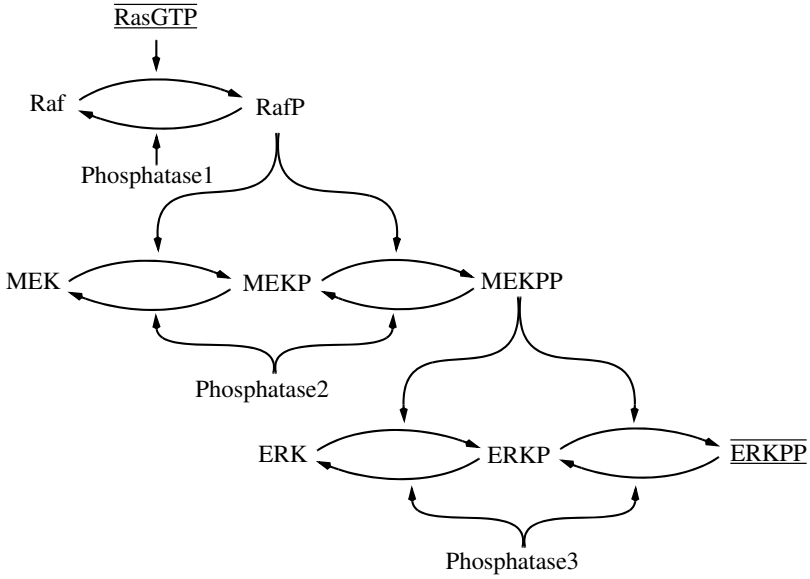


Fig. 3. The general scheme of the considered signalling pathway: a three-stage double phosphorylation cascade. Each phosphorylation/dephosphorylation step applies the mass action kinetics pattern $A+E \rightleftharpoons AE \rightarrow B+E$. We consider RasGTP as the input signal and ERKPP as the output signal.

4 The Qualitative Approach

4.1 Qualitative Modelling

To allow formal reasoning of the general scheme of a signal transduction cascade, which is given in Figure 3 in an informal way, we are going to derive a corresponding Petri net. Petri nets enjoy formal semantics amenable to mathematically sound analysis techniques. The first two definitions introduce the standard notion of place/transition Petri nets, which represents the basic class in the ample family of Petri net models.

Definition 1 (Petri net, Syntax). *A* Petri net *is a quadruple* $\mathcal{N} = (P, T, f, m_0)$, *where*

- P *and* T *are finite, non empty, and disjoint sets.* P *is the set of* places *(in the figures represented by circles).* T *is the set of* transitions *(in the figures represented by rectangles).*
- $f : ((P \times T) \cup (T \times P)) \to \mathbb{N}_0$ *defines the set of directed* arcs*, weighted by nonnegative integer values.*
- $m_0 : P \to \mathbb{N}_0$ *gives the* initial marking*.*

Thus, Petri nets (or nets for short) are weighted, directed, bipartite graphs. The idea to use Petri nets for the representation of biochemical networks is rather intuitive and has been mentioned by Carl Adam Petri himself in one of his internal research reports on interpretation of net theory in the seventies. It has also been used as the very first introductory example in [Mur89], and we follow that idea in this tutorial, compare Figure 4.

Places usually model passive system components like conditions, species or any kind of chemical compounds, e.g. proteins or proteins complexes, playing the role of precursors or products. Transitions stand for active system components like atomic actions or any kind of chemical reactions, e.g. association, disassociation, phosphorylation, or dephosphorylation, transforming precursors into products.

The arcs go from precursors to reactions (ingoing arcs), and from reactions to products (outgoing arcs). In other words, the preplaces of a transition correspond to the reaction's precursors, and its postplaces to the reaction's products. Enzymes establish side conditions and are connected in both directions with the reaction they catalyse; we get a read arc.

Arc weights may be read as the multiplicity of the arc, reflecting known stoichiometries. Thus, the (pseudo) arc weight 0 stands for the absence of an arc. The arc weight 1 is the default value and is usually not given explicitly.

A place carries an arbitrary number of *tokens*, represented as black dots or a natural number. The number zero is the default value and usually not given explicitly. Tokens can be interpreted as the available amount of a given species in number of molecules or moles, or any abstract, i.e. discrete concentration level.

In the most abstract way, a concentration can be thought of as being 'high' or 'low' (present or absent). Generalizing this Boolean approach, any continuous concentration range can be divided into a finite number of equally sized subranges (equivalence classes), so that the concentrations within can be considered to be equivalent. The current number of tokens on a place will then specify the current level of the species' concentration, e.g. the absence of tokens specifies level 0. In the following, when speaking in terms of level semantics, we always give the highest level number.

A particular arrangement of tokens over the places of the net is called a marking, modelling a system state. In this paper, the notions *marking* and *state* are used interchangeably.

We introduce the following notions and notations. $m(p)$ yields the number of tokens on place p in the marking m. A place p with $m(p) = 0$ is called *clean (empty, unmarked)* in m, otherwise it is called *marked* (non-clean). A set of places is called clean if all its places are clean, otherwise marked. The preset of a node $x \in P \cup T$ is defined as ${}^\bullet x := \{y \in P \cup T | f(y, x) \neq 0\}$, and its postset as $x^\bullet := \{y \in P \cup T | f(x, y) \neq 0\}$. Altogether we get four types of sets:

- ${}^\bullet t$, the preplaces of a transition t, consisting of the reaction's precursors,
- $t^\bullet$, the postplaces of a transition t, consisting of the reaction's products,
- ${}^\bullet p$, the pretransitions of a place p, consisting of all reactions producing this species,
- $p^\bullet$, the posttransitions of a place p, consisting of all reactions consuming this species.

We extend both notions to a set of nodes $X \subseteq P \cup T$ and define the set of all prenodes ${}^\bullet X := \bigcup_{x \in X} {}^\bullet x$, and the set of all postnodes $X^\bullet := \bigcup_{x \in X} x^\bullet$. See Figure 11 for an illustration of these notations.

Petri net, Semantics Up to now we have introduced the static aspects of a Petri net only. The behaviour of a net is defined by the firing rule, which consists of two parts: the precondition and the firing itself.

Definition 2 (Firing rule). *Let $\mathcal{N} = (P, T, f, m_0)$ be a Petri net.*

- *A transition t is* enabled *in a marking m, written as $m[t\rangle$, if $\forall p \in {}^\bullet t : m(p) \geq f(p, t)$, else* disabled.
- *A transition t, which is enabled in m, may fire.*
- *When t in m* fires*, a new marking m' is reached, written as $m[t\rangle m'$, with $\forall p \in P : m'(p) = m(p) - f(p, t) + f(t, p)$.*
- *The firing happens atomically and does not consume any time.*

Please note, a transition is never forced to fire. Figuratively, the firing of a transition moves tokens from its preplaces to its postplaces, while possibly changing the number of tokens, compare Figure 4. Generally, the firing of a transition changes the formerly current marking to a new reachable one, where some transitions are not enabled anymore while others get enabled. The repeated firing of transitions establishes the behaviour of the net.

The whole net behaviour consists of all possible partially ordered firing sequences (partial order semantics), or all possible totally ordered firing sequences (interleaving semantics), respectively.

Every marking is defined by the given token situation in all places $m \in \mathbb{N}_0^{|P|}$, whereby $|P|$ denotes the number of places in the Petri net. All markings, which can be reached from a given marking m by any firing sequence of arbitrary length, constitute the *set of reachable markings* $[m\rangle$. The set of markings $[m_0\rangle$ reachable from the initial marking is said to be the *state space* of a given system.

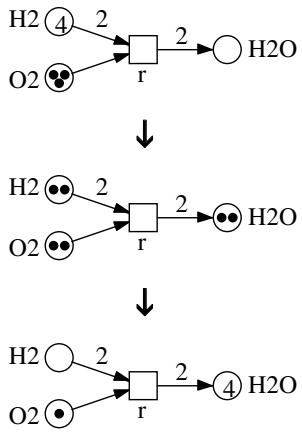


Fig. 4. The Petri net for the well known chemical reaction r: $2H_2 + O_2 \rightarrow 2H_2O$ and three of its markings (states), connected each by a firing of the transition r. The transition is not enabled anymore in the marking reached after these two single firing steps.

All notions introduced in the following in this section refer to a place/transition Petri net according to Definitions 1 and 2.

Running example In this modelling spirit we are now able to create a Petri net for our running example. We start with building blocks for some typical chemical reaction equations as shown in Figure 5. We get the Petri net in Figure 6 for our running example by composing these building blocks according to the scheme of Figure 3. As we will see later, this net structure corresponds exactly to the set of ordinary differential equations given in [LBS00]. Thus, the net can equally be derived by SBML import and automatic layout, manually improved from this ODE model.

Reversible reactions have to be modelled explicitly by two opposite transitions. However in order to retain the elegant graph structure of Figure 6, we use macro transitions, each of which stands here for a reversible reaction. The entire (flattened) place/transition Petri net consists of 22 places and 30 transitions, where $r1, r2, \ldots$ stand for reaction (transition) labels.

We associate a discrete concentration with each of the 22 species. In the qualitative analysis we apply Boolean semantics where the concentrations can be thought of as being "high" or "low" (above or below a certain threshold). This results into a two level model, and we extend this to a multi-level model in the quantitative analysis, where each discrete level stands for an equivalence class of possibly infinitely many concentrations. Then places can be read as integer variables.

4.2 Qualitative Analysis

A preliminary step will usually execute the net, which allows us to experience the model behaviour by following the token flow[1]. Having established initial confidence in the model by playing the token game, the system needs to be

[1] If the reader would like to give it a try, just download our Petri net tool [Sno08].

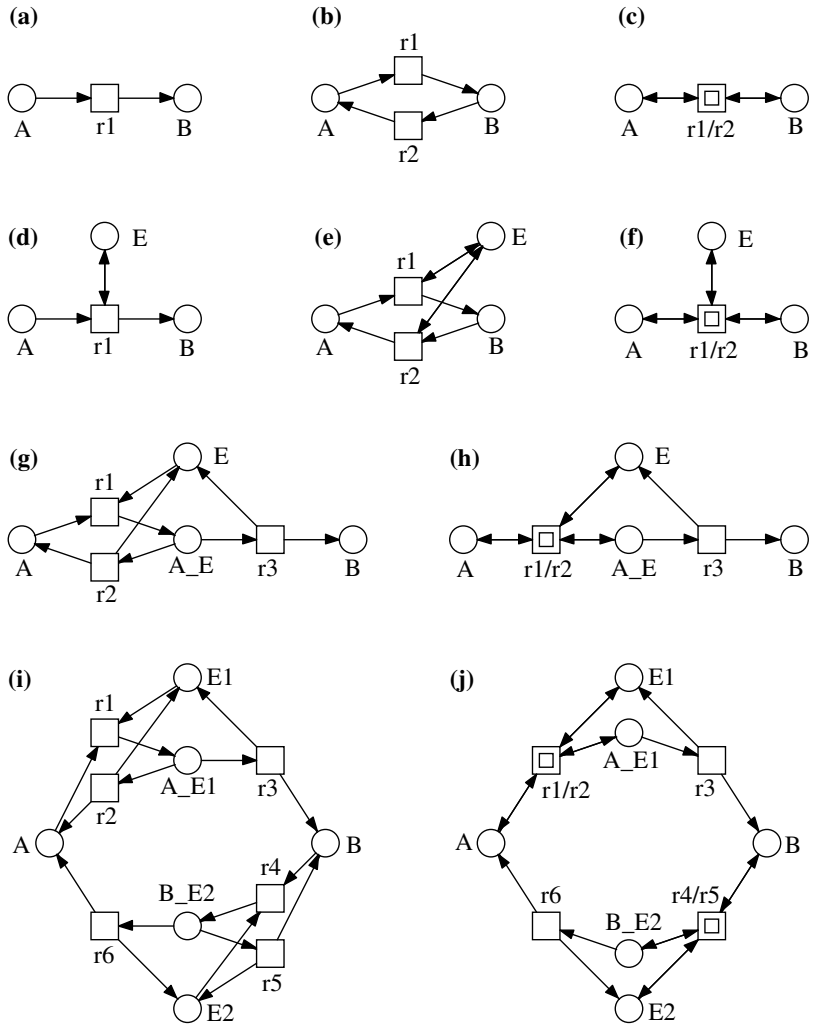
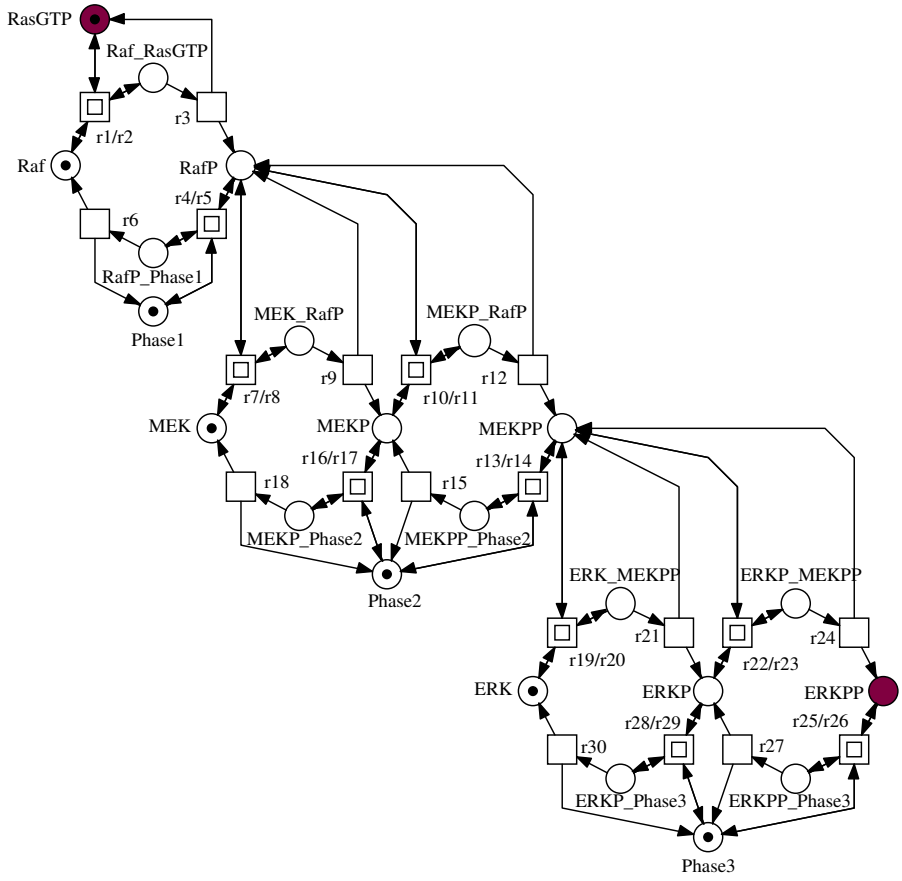


Fig. 5. The Petri net components for some typical basic structures of biochemical reaction networks. **(a)** simple reaction $A \rightarrow B$; **(b)** reversible reaction $A \rightleftharpoons B$; **(c)** hierarchical notation of (b); **(d)** simple enzymatic reaction, Michaelis-Menten kinetics; **(e)** reversible enzymatic reaction, Michaelis-Menten kinetics; **(f)** hierarchical notation of (e); **(g)** enzymatic reaction, mass action kinetics, $A + E \rightleftharpoons A\_E \rightarrow B + E$; **(h)** hierarchical notation of (g); **(i)** two enzymatic reactions, mass action kinetics, building a cycle; **(j)** hierarchical notation of (i). Two concentric squares are macro transitions, allowing the design of hierarchical net models. They are used here as shortcuts for reversible reactions. Two opposite arcs denote read arcs, see (d) and (e), establishing side conditions for a transition's firing.




```
PUR ORD HOM NBM CSV SCF CON SC  FT0 TF0 FP0 PF0 NC
Y   Y   Y   Y   N   N   Y   Y   N   N   N   N   nES
DTP CPI CTI SCTI SB k-B 1-B DCF DSt DTr LIV REV
Y   Y   Y   N   Y   Y   Y   N   0   N   Y   Y
```

Fig. 6. The bipartite graph for the extended ERK pathway model according to the scheme in Figure 3. Places (circles) stand for species (proteins, protein complexes). Protein complexes are indicated by an underscore "_" between the constituent protein names. The suffixes P or PP indicate phosphorylated or doubly phosphorylated forms respectively. The name *Phase* serves as shortcut for *Phosphatase*. The species that are read as input/output signals are given in grey. Transitions (squares) stand for irreversible reactions, while macro transitions (two concentric squares) specify reversible reactions, compare Figure 5. The initial state is systematically constructed using standard Petri net analysis techniques. At the bottom the two-line result vector as produced by Charlie [Cha08] is given. Properties of interest in this vector for this biochemical network are explained in the text.

formally analysed. Formal analyses are exhaustive, opposite to the token game, which exemplifies the net behaviour.

(0) General behavioural properties The first step in analysing a Petri net usually aims at deciding general behavioural properties, i.e. properties which can be formulated independently from the special functionality of the network under consideration. There are basically three of them, which are orthogonal: boundedness, liveness, and reversibility [Mur89]. We start with an informal characterisation of the key issues.

- *boundedness* For every place it holds that: Whatever happens, the maximal number of tokens on this place is bounded by a constant. This precludes overflow by unlimited increase of tokens.
- *liveness* For every transition it holds that: Whatever happens, it will always be possible to reach a state where this transition gets enabled. In a live net, all transitions are able to contribute to the net behaviour forever, which precludes dead states, i.e. states where none of the transitions are enabled.
- *reversibility* For every state it holds that: Whatever happens, the net will always be able to reach this state again. So the net has the capability of self-reinitialization.

In most cases these are requirable properties. To be precise, we give the following formal definitions, elaborating these notions in more details.

Definition 3 (Boundedness).

- *A place p is* k-bounded *(bounded for short) if there exists a positive integer number k, which represents an upper bound for the number of tokens on this place in all reachable markings of the Petri net:*
  $\exists\ k \in \mathbb{N}_0 : \forall m \in [m_0\rangle : m(p) \leq k$.
- *A Petri net is* k-bounded *(bounded for short) if all its places are* k-bounded.
- *A Petri net is* structurally bounded *if it is bounded in any initial marking.*

Definition 4 (Liveness of a transition).

- *A transition t is* dead in the marking m *if it is not enabled in any marking m' reachable from m:*
  $\nexists\ m' \in [m\rangle : m'[t\rangle$.
- *A transition t is* live *if it is not dead in any marking reachable from m_0.*

Definition 5 (Liveness of a Petri net).

- *A marking m is* dead *if there is no transition which is enabled in m.*
- *A Petri net is* deadlock-free (weakly live) *if there are no reachable dead markings.*
- *A Petri net is* live (strongly live) *if each transition is live.*

Definition 6 (Reversibility). *A Petri net is* reversible *if the initial marking can be reached again from each reachable marking:* $\forall m \in [m_0\rangle : m_0 \in [m\rangle$.

Finally we introduce the general behavioural property *dynamic conflict*, which refers to a marking enabling two transitions, but the firing of one transition disables the other one. The occurrence of dynamic conflicts causes alternative (branching) system behaviour, whereby the decision between these alternatives is taken nondeterministically. See Figure 7 for an illustration of these behavioural properties.

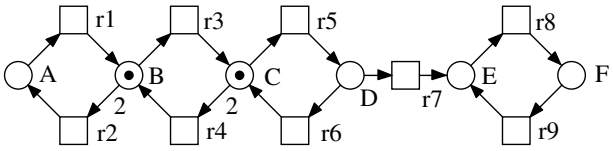


Fig. 7. A net to illustrate the general behavioural properties. The place A is 0-bounded, place B is 1-bounded and all other places are 2-bounded, so the net is 2-bounded. The transitions r1 and r2 in the leftmost cycle are dead at the initial marking. The transitions r8 and r9 in the rightmost cycle are live. All other transitions are not live; so the net is weakly live. The net is not reversible, because there is no counteraction to the token decrease by firing of r4. There are dynamic conflicts, e.g. between r4 and r5 in a marking with m(C)=2.

Running example Our net enjoys the three orthogonal general properties of a qualitative Petri net: it is bounded, even structural bounded (SB), live (LIV), and reversible (REV).

Boundedness can always be decided in a static way, i.e. without construction of the state space, while the remaining behavioural properties generally require dynamic analysis techniques, i.e. the explicit construction of the partial or full state space. However as we will see later, freedom of dead states (DSt) can still be decided in a static way for our running example.

The essential steps of the systematic analysis procedure for our running example are given in more detail as follows. They represent a typical pattern how to proceed. So they may be taken as a recipe how to analyse your own system.

(1) Structural properties The following structural properties are elementary graph properties and reflect the modelling approach. They can be read as preliminary consistency checks to preclude production faults in drawing the net. Remarkably, certain combinations of structural properties allow conclusions on behavioural properties; some examples of such conclusions will be mentioned. The list follows the order as used in the two-line result vector produced by our qualitative analysis tool Charlie [Cha08], compare Figure 6.

PUR A Petri net is *pure* if
$\forall x, y \in P \cup T : f(x, y) \neq 0 \Rightarrow f(y, x) = 0$,
i.e. there are no two nodes, connected in both directions. This precludes read arcs. Then the net structure is fully represented by the incidence matrix, which is used for the calculation of the P- and T-invariants, see step (2).

ORD A Petri net is *ordinary* if
$\forall x, y \in P \cup T : f(x, y) \neq 0 \Rightarrow f(x, y) = 1$,
i.e. all arc weights are equal to 1. This includes homogeneity. A non-ordinary Petri net cannot be live and 1-bounded at the same time.

HOM A Petri net is *homogeneous* if
$\forall p \in P : t, t' \in p^\bullet \Rightarrow f(p, t) = f(p, t')$,
i.e. all outgoing arcs of a given place have the same multiplicity.

NBM A net has *non-blocking multiplicity* if
$\forall p \in P : {}^\bullet p \neq \emptyset \wedge min\{f(t, p) | \forall t \in {}^\bullet p\} \geq max\{f(p, t) | \forall t \in p^\bullet\}$,
i.e. an input place causes blocking multiplicity. Otherwise, it must hold for each place: the minimum of the multiplicities of the incoming arcs is not less than the maximum of the multiplicities of the outgoing arcs.

CSV A Petri net is *conservative* if
$\forall t \in T : \sum_{p \in {}^\bullet t} f(p, t) = \sum_{p \in t^\bullet} f(t, p)$,
i.e. all transitions add exactly as many tokens to their postplaces as they subtract from their preplaces, or briefly, all transitions fire token-preservingly. A conservative Petri net is structurally bounded.

SCF A Petri net is *static conflict free* if
$\forall t, t' \in T : t \neq t' \Rightarrow {}^\bullet t \cap {}^\bullet t' = \emptyset$,
i.e. there are no two transitions sharing a preplace. Transitions involved in a static conflict compete for the tokens on shared preplaces. Thus, static conflicts indicate situations where dynamic conflicts, i.e. nondeterministic choices, may occur in the system behaviour. However, it depends on the token situation whether a conflict does actually occur dynamically. There is no nondeterminism in SCF nets.

CON A Petri net is *connected* if it holds for every two nodes a and b that there is an undirected path between a and b. Disconnected parts of a Petri net cannot influence each other, so they can usually be analysed separately. In the following we consider only connected Petri nets.

SC A Petri net is *strongly connected* if it holds for every two nodes a and b that there is a directed path from a to b. Strong connectedness involves connectedness and the absence of boundary nodes. It is a necessary condition for a Petri net to be live and bounded at the same time.

FT0, TF0, FP0, PF0 A node $x \in P \cup T$ is called *boundary node* if
${}^\bullet x = \emptyset \vee x^\bullet = \emptyset$. Boundary nodes exist in four types:
- input transition - a transition without preplaces (${}^\bullet t = \emptyset$, shortly FT0),
- output transition - a transition without postplaces ($t^\bullet = \emptyset$, shortly TF0),
- input place - a place without pretransitions (${}^\bullet p = \emptyset$, shortly FP0),
- output place - a place without posttransitions ($p^\bullet = \emptyset$, shortly PF0).

A net with boundary nodes cannot be bounded and live at the same time. For example, an input transition is always enabled, so its postplaces are

unbounded, while input places preclude liveness. Boundary nodes model interconnections of an open system with its environment. A net without boundary nodes is self-contained, i.e. a closed system. It needs a non-clean initial marking to become live.

Definition 7 (Net structure classes).

- *A Petri net is called* State Machine *(SM) if*
  $\forall t \in T : |{}^\bullet t| = |t^\bullet| \leq 1$,
  i.e. there are neither forward branching nor backward branching transitions.
- *A Petri net is called* Synchronisation Graph *(SG) if*
  $\forall p \in P : |{}^\bullet p| = |p^\bullet| \leq 1$,
  i.e. there are neither forward branching nor backward branching places.
- *A Petri net is called* Extended Free Choice *(EFC) if*
  $\forall p, q \in P : p^\bullet \cap q^\bullet = \emptyset \vee p^\bullet = q^\bullet$,
  i.e. transitions in conflict have identical sets of preplaces.
- *A Petri net is called* Extended Simple *(ES) if*
  $\forall p, q \in P : p^\bullet \cap q^\bullet = \emptyset \vee p^\bullet \subseteq q^\bullet \vee q^\bullet \subseteq p^\bullet$,
  i.e. every transition is involved in one conflict at most.

Please note, these definitions refer to the net structure only, neglecting any arc multiplicities. However, these net classes are especially helpful in the setting of ordinary nets. SM and SG [2] are dual notions; a SM net can be converted into an SG net by exchanging places and transitions, and vice versa. Both net classes are properly included in the EFC net class, which again is properly included in the ES net class.

SM nets are conservative, and thus the prototype of bounded models; they correspond to the well-known notion of finite state automata. SG nets are free of static conflicts, and therefore of nondeterminism. In EFC nets, transitions in conflict are always together enabled or disabled; so there is always a free choice between them in dynamic conflict situations. EFC nets have the pleasant property of monotonous liveness, i.e. if they are live in the marking m, then they remain live for any other marking m' with $m' \geq m$. In ES nets, the conflict relation is transitive: if $t1$ and $t2$ are in conflict, and $t2$ and $t3$ are in conflict, then $t1$ and $t3$ are in conflict too. ES nets have the distinguished property to be live independent of time, i.e. if they are live, then they remain live under any timing [Sta89].

All these structural properties do not depend on the initial marking. Most of these properties can be locally decided in the graph structure. Connectedness and strong connectedness need to consider the global graph structure, which can be done using standard graph algorithms.

Running example The net is pure and ordinary, therefore homogeneous as well, but not conservative. There are static conflicts. The net structure does

[2] We use *synchronisation graph* instead of the more popular term *marked graph*, which might cause confusion.

not comply to any of the introduced net structure classes, so it is said to be *not Extended Simple* (nES). The net is strongly connected, which includes connectedness and absence of boundary nodes, and thus self-contained, i.e. a closed system. Therefore, in order to make the net live, we have to construct an initial marking, see step (3) below.

(2) Static decision of marking-independent behavioural properties To open the door to analysis techniques based on linear algebra, we represent the net structure by a matrix, called incidence matrix in the Petri net community, and stoichiometric matrix in systems biology. We briefly recall the essential technical terms.

Definition 8 (P-invariants, T-invariants).

- *The* incidence matrix *of* $\mathcal{N}$ *is a matrix* $\mathbb{C} : P \times T \to \mathbb{Z}$*, indexed by* P *and* T*, such that* $\mathbb{C}(p,t) = f(t,p) - f(p,t)$*.*
- *A* place vector (transition vector) *is a vector* $x : P \to \mathbb{Z}$*, indexed by* P *(*$y : T \to \mathbb{Z}$*, indexed by* T*).*
- *A place vector (transition vector) is called a* P-invariant (T-invariant) *if it is a nontrivial nonnegative integer solution of the linear equation system* $x \cdot \mathbb{C} = 0$ *(*$\mathbb{C} \cdot y = 0$*).*
- *The set of nodes corresponding to an invariant's nonzero entries are called the* support *of this invariant* x*, written as* $supp\,(x)$*.*
- *An invariant* x *is called* minimal *if* $\nexists$ *invariant* $z : supp\,(z) \subset supp\,(x)$*, i.e. its support does not contain the support of any other invariant* z*, and the greatest common divisor of all nonzero entries of* x *is 1.*
- *A net is* covered by P-invariants, shortly CPI, (covered by T-invariants, shortly CTI) *if every place (transition) belongs to a P-invariant (T-invariant).*

CPI causes structural boundedness (SB), i.e. boundedness for any initial marking. CTI is a necessary condition for bounded nets to be live. But maybe even more importantly, invariants are a beneficial technique in model validation, and the challenge is to check all invariants for their biological plausibility. Therefore, let's elaborate these notions more carefully, compare also Figure 8.

The *incidence matrix* of a Petri net is an integer matrix $\mathbb{C}$ with a row for each place and a column for each transition. A matrix entry $\mathbb{C}(p,t)$ gives the token change on place p by the firing of transition t. Thus, a preplace of t, which is not a postplace of t, has a negative entry, while a postplace of t, which is not a preplace of t, has a positive entry, each corresponding to the arc multiplicities. The entry for a place, which is preplace as well as postplace of a transition, gives the difference of the multiplicities of the transition's outgoing arc minus the transition's ingoing arc. In this case we lose information; the non-ordinary net structure cannot be reconstructed uniquely out of the incidence matrix.

The columns of $\mathbb{C}$ are place vectors, i.e. vectors with as many entries as there are places, describing the token change on a marking by the firing of the transition defining the column index. The rows of $\mathbb{C}$ are transition vectors, i.e. vectors

with as many entries as there are transitions, describing the influence of all transitions on the tokens in the place, defining the row index. For stoichiometric reaction networks, e.g. metabolic networks, the incidence matrix coincides with the stoichiometric matrix.

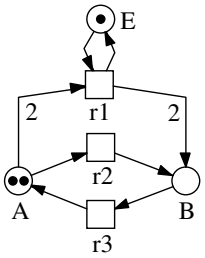
A *P-invariant* x is a nonzero and nonnegative integer place vector such that $x \cdot \mathbb{C} = 0$; in words, for each transition it holds that: multiplying the P-invariant with the transition's column vector yields zero. Thus, the total effect of each transition on the P-invariant is zero, which explains its interpretation as a token conservation component. A P-invariant stands for a set of places over which the weighted sum of tokens is constant and independent of any firing, i.e. for any markings m_1, m_2, which are reachable by the firing of transitions, it holds that $x \cdot m_1 = x \cdot m_2$. In the context of metabolic networks, P-invariants reflect substrate conservations, while in signal transduction networks P-invariants often correspond to the several states of a given species (protein or protein complex). A place belonging to a P-invariant is obviously bounded.

Analogously, a *T-invariant* y is a nonzero and nonnegative integer transition vector such that $\mathbb{C} \cdot y = 0$; in words, for each place it holds that: multiplying the place's row with the T-invariant yields zero. Thus, the total effect of the T-invariant on a marking is zero. A T-invariant has two interpretations in the given biochemical context.

- The entries of a T-invariant specify a multiset of transitions which by their partially ordered firing reproduce a given marking, i.e. basically occurring one after the other. This partial order sequence of the T-invariant's transitions may contribute to a deeper understanding of the net behaviour. A T-invariant is called *feasible* if such a behaviour is actually possible in the given marking situation.
- The entries of a T-invariant may also be read as the relative firing rates of transitions, all of them occurring permanently and concurrently. This activity level corresponds to the steady state behaviour.

r1: 2 A $\xrightarrow{E}$ 2 B
r2/3: A $\rightleftharpoons$ B

	r1	r2	r3
A	–2	–1	1
B	2	1	–1
E	0	0	0



$x_1 = (1, 1, 0) = (\text{A, B}),$
$x_2 = (0, 0, 1) = (\text{E})$

$y_1 = (1, 0, 2) = (\text{r1, } 2 \cdot \text{r3}),$
$y_2 = (0, 1, 1) = (\text{r2, r3})$

$y_3 = (1, 1, 3) = y_1 + y_2$

Fig. 8. Two reaction equations with the corresponding Petri net, its incidence matrix, and the minimal P-invariants x_1, x_2, and the minimal T-invariants y_1, y_2, and a non-minimal T-invariant y_3. The invariants are given in the standard vector notation as well as in a shorthand notation, listing the nonzero entries only. The net is not pure; the incidence matrix does not reflect the dependency of r1 on E.

The two transitions modelling the two directions of a reversible reaction always make a minimal T-invariant; thus they are called *trivial T-invariants*. A net which is covered by nontrivial T-invariants is said to be *strongly covered by T-invariants* (SCTI). Transitions not covered by nontrivial T-invariants are candidates for model reduction, e.g. if the model analysis is concerned with steady state analysis only.

The set x_i of all minimal P-invariants (T-invariants) of a given net is unique and represents a generating system for all P-invariants (T-invariants). All invariants x can be computed as nonnegative linear combinations: $n \cdot x = \sum(a_i \cdot x_i)$, with $n, a_i \in \mathbb{N}_0$, i.e., the allowed operations are addition, multiplication by a natural number, and division by a common divisor.

Technically, we need to solve a homogenous linear equation system over natural numbers (nonnegative integers). This restriction of the data space establishes – from a mathematical point of view – a challenge, so there is no closed formula to compute the solutions. However there are algorithms – actually, a class of algorithms – constructing the solution (to be precise: the generating system for the solution space) by systematically considering all possible candidates.

This algorithm class has been repetitively re-invented over the years. So, these algorithms come along with different names; but if you take a closer look, you will always encounter the same underlying idea. All these versions may be classified as "positive Gauss elimination"; the incidence matrix is systematically transformed to a zero matrix by suitable matrix operations.

However, there are net structures where we get the invariants almost for free. For ordinary state machines it holds:

- each (minimal) cycle is a (minimal) T-invariant;
- for strongly connected state machines, the reverse direction holds also: each (minimal) T-invariant corresponds to a (minimal) cycle;
- all places of a strongly connected state machine form a minimal P-invariant.

Likewise, for ordinary synchronisation graphs it holds:

- each (minimal) cycle is a (minimal) P-invariant;
- for strongly connected synchronisation graphs, the reverse direction holds also: each (minimal) P-invariant corresponds to a (minimal) cycle;
- all transitions of a strongly connected synchronisation graph form a minimal T-invariant.

A minimal P-invariant (T-invariant) defines a connected subnet, consisting of its support, its pre- and posttransitions (pre- and postplaces), and all arcs in between. There are no structural limitations for such subnets induced by minimal invariants, compare Figure 9, but they are always connected, however not necessarily strongly connected. These minimal self-contained subnets may be read as a decomposition into token preserving or state repeating modules, which should have an enclosed biological meaning. However, minimal invariants generally overlap, and in the worst-case there are exponentially many of them.

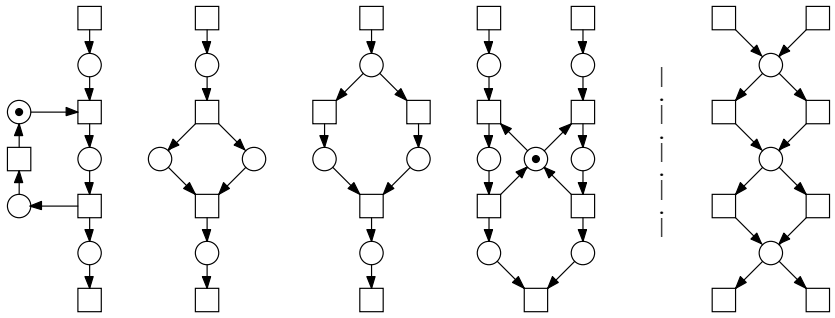


Fig. 9. The four nets on the left are each covered by one minimal T-invariants. Invariants can contain any structures (from left to right): cycles, forward/backward branching transitions, forward branching places, backward branching places. Generally, invariants overlap, and in the worst-case there are exponentially many of them; the net on the far-right has 2^4 T-invariants.

Running example There are seven minimal P-invariants covering the net (CPI), and consequently the net is bounded for any initial marking (SB). All these P-invariants x_i contain only entries of 0 and 1, permitting a shorthand specification by just giving the names of the places involved.

$x_1 =$ (**RasGTP,** Raf_RasGTP)
$x_2 =$ (**Raf,** Raf_RasGTP, RafP, RafP_Phase1, MEK_RafP, MEKP_RafP)
$x_3 =$ (**MEK,** MEK_RafP, MEKP_RafP, MEKP_Phase2, MEKPP_Phase2, ERK_MEKPP, ERKP_MEKPP, MEKPP, MEKP)
$x_4 =$ (**ERK,** ERK_MEKPP, ERKP_MEKPP, ERKP, ERKPP_Phase3, ERKP_Phase3, ERK_PP)
$x_5 =$ (**Phase1,** RafP_Phase1)
$x_6 =$ (**Phase2,** MEKP_Phase2, MEKPP_Phase2)
$x_7 =$ (**Phase3,** ERKP_Phase3, ERKPP_Phase3)

Each P-invariant stands for a reasonable conservation rule, the species preserved being given by the first name in the invariant. Due to the chosen naming convention, this particular name also appears in all the other place names of the same P-invariant.

The net under consideration is also covered by T-invariants (CTI), however not strongly covered (SCTI). Besides the expected ten trivial T-invariants for the ten reversible reactions, there are five nontrivial, but obvious minimal T-invariants, each corresponding to one of the five phosphorylation/dephosphorylation cycles in the network structure:

$y_1 =$ (r1, r3, r4, r6),
$y_2 =$ (r7, r9, r16, r18),
$y_3 =$ (r10, r12, r13, r15),
$y_4 =$ (r19, r21, r28, r30),
$y_5 =$ (r22, r24, r25, r27).

The interesting net behaviour, demonstrating how input signals finally cause output signals, is contained in a nonnegative linear combination of all five nontrivial T-invariants,

$$y_{1-5} = y_1 + y_2 + y_3 + y_4 + y_5,$$

which is called an I/O T-invariant in the following. The I/O T-invariant is systematically constructed by starting with the two minimal T-invariants, involving the input and output signal, which define disconnected subnetworks. Then we add minimal sets of minimal T-invariants to get a connected subnet, which corresponds to a T-invariant feasible in the initial marking. For our running example, the solution is unique, which is not generally the case.

The automatic identification of nontrivial minimal T-invariants is in general useful as a method to highlight important parts of a network, and hence aids its comprehension by biochemists, especially when the entire network is too complex to easily comprehend.

P/T-invariants relate only to the structure, i.e. they are valid independently of the initial marking. In order to proceed we first need to generate an initial marking.

(3) Initial marking construction For a systematic construction of the initial marking, we consider the following criteria.

- Each P-invariant needs at least one token.
- All (nontrivial) T-invariants should be feasible, meaning, the transitions, making up the T-invariant's multi-set can actually be fired in an appropriate (partial) order.
- Additionally, it is common sense to look for a minimal marking (as few tokens as possible), which guarantees the required behaviour.
- Within a P-invariant, choose the species with the most *inactive* or the *monomeric* state.

Running example Taking all these criteria together, the initial marking on hand is: RasGTP, MEK, ERK, Phase1, Phase2 and Phase3 get each one token, while all remaining places are empty. With this initial marking, the net is covered by 1-P-invariants (exactly one token in each P-invariant), therefore the net is 1-bounded (indicated as 1-B in the analysis result vector, compare Figure 6). That is in perfect accordance with the understanding that in signal transduction networks a P-invariant comprises all the different states of one species. Obviously, each species can be only in one state at any time.

Generalising this reasoning to a multi-level concept, we could assign n tokens to each place representing the most inactive state, in order to indicate the highest concentration level for them in the initial state. The "abstract" mass conservation within each P-invariant would then be n tokens, which could be distributed fairly freely over the P-invariant's places during the behaviour of the

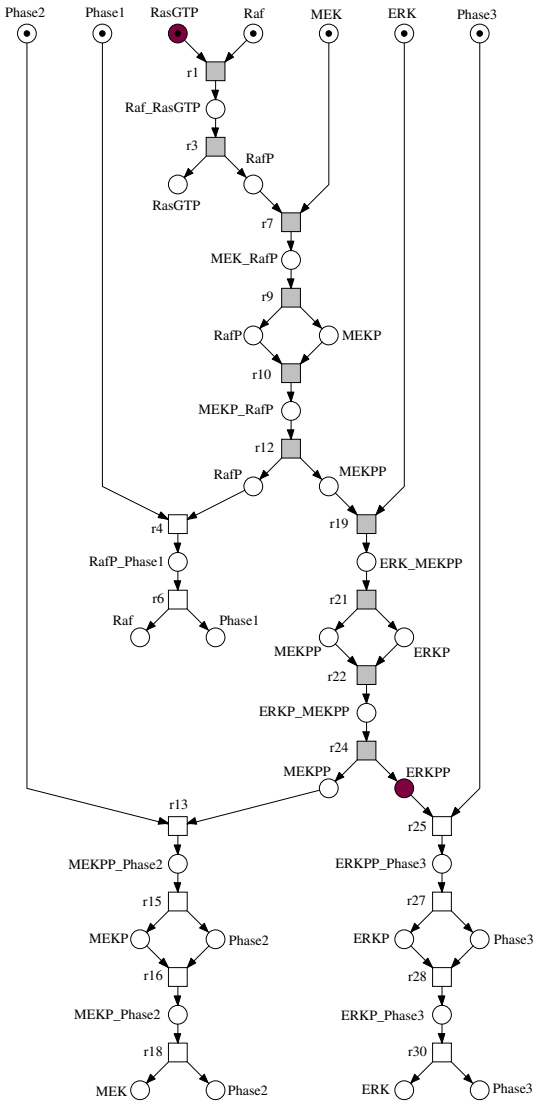


Fig. 10. The beginning of the infinite partial order run of the I/O T-invariant $y_{1-5} = y_1 + y_2 + y_3 + y_4 + y_5$ of the place/transition Petri net given in Figure 6. We get this run by unfolding the behaviour of the subnet induced by the T-invariant, whereby any concurrency is preserved. Here, transitions represent events, labelled by the name of the reaction taking place, while places stand for binary conditions, labelled by the name of the species, set or reset by the event, respectively. The highlighted set of transitions and places is the required minimal sequence of events to produce the output signal ERKPP. We get a totally ordered sequence of events for our running example. Generally, this sequence will be partially ordered only.

model. This results in a dramatic increase of the state space, as we will later see, while not improving the qualitative reasoning.

We check the I/O T-invariant for feasibility in the constructed initial marking, which then involves the feasibility of all trivial T-invariants. In order to preserve all the concurrency information we have, we construct a new net which describes the behaviour of our system net under investigation. We obtain an *infinite partial order run*, the beginning of which is given as labelled condition/event net in Figure 10. Here, transitions represent events, labelled by the name of the reaction taking place, while places stand for binary conditions, labelled by the name of the species, set or reset by the event, respectively. We get this run by unfolding the behaviour of the subnet induced by the T-invariant. This run can be characterized in a shorthand notation by the following set of partially ordered words out of the alphabet of all transition labels T (";" stands for "sequentiality", "‖" for "concurrency"):

(r1; r3; r7; r9; r10; r12;
((r4; r6) ‖
((r19; r21; r22; r24);
((r13; r15; r16; r18) ‖ (r25; r27; r28; r30))))).

This partial order run gives further insight into the dynamic behaviour of the network, which may not be apparent from the standard net representation, e.g. we are able to follow the (minimal) producing process of the proteins RafP, MEKP, MEKPP, ERKP and ERKPP (highlighted in Figure 10), and we notice the clear independence, i.e. concurrency of the dephosphorylation in all three levels. The entire run describes the whole network behaviour triggered by the input signal, i.e. including the dephosphorylation. This unfolding is completely defined by the net structure, the initial marking and the multiset of firing transitions. Thus it can be constructed automatically.

Having established and justified our initial marking, we proceed to the next steps of the analysis.

(4) Static decision of marking-dependent behavioural properties The following advanced structural Petri net properties can be decided by combinatorial algorithms. First, we need to introduce two new notions.

Definition 9 (Structural deadlocks, traps).

- *A nonempty set of places $D \subseteq P$ is called* structural deadlock (co-trap) *if $^\bullet D \subseteq D^\bullet$ (the set of pretransitions is contained in the set of posttransitions), i.e. every transition which fires tokens onto a place in this structural deadlock set, also has a preplace in this set.*
- *A set of places $Q \subseteq P$ is called* trap *if $Q^\bullet \subseteq {}^\bullet Q$ (the set of posttransitions is contained in the set of pretransitions), i.e. every transition which subtracts tokens from a place of the trap set, also has a postplace in this set.*

Pretransitions of a structural deadlock [3] cannot fire if the structural deadlock is clean. Therefore, a structural deadlock cannot get tokens again as soon as it is clean, and then all its posttransitions $t \in D^\bullet$ are dead. A Petri net without structural deadlocks is live, while a system in a dead state has a clean structural deadlock.

Posttransitions of a trap always return tokens to the trap. Therefore, once a trap contains tokens, it cannot become clean again. There can be a decrease of the total token amount within a trap, but not down to zero.

An input place p establishes a structural deadlock $D = \{p\}$ on its own, and an output place q, a trap $Q = \{q\}$. If each transition has a preplace, then $P^\bullet = T$, and if each transition has a postplace, then $^\bullet P = T$. Therefore, in a net without boundary transitions, the whole set of places is a structural deadlock as well as a trap. If D and D' are structural deadlocks (traps), then $D \cup D'$ is also a structural deadlock (trap).

A structural deadlock (trap) is *minimal* if it does not properly contain a structural deadlock (nonempty trap). The network, defined by a minimal structural deadlock (trap), is strongly connected. A trap is *maximal* if it is not a proper subset of a trap. Every structural deadlock includes a unique maximal trap with respect to set inclusion (which may be empty).

The support of a P-invariant is structural deadlock and trap at the same time. But caution: not every place set which is a structural deadlock as well as a trap is a P-invariant. Even more, a P-invariant may properly contain a structural deadlock. Of special interest are often those minimal deadlocks (traps), which are not at the same time a P-invariant, for which we introduce the notion *proper deadlock (trap)*. See also Figure 11 for an example to illustrate these two notions of structural deadlock and trap.

Structural deadlock and trap are closely related but contrasting notions. When they come on their own, we get usually deficient behaviour. However, both notions have the power to complement each other perfectly.

Definition 10 (Deadlock trap property).
A Petri net satisfies the deadlock trap property (DTP) if

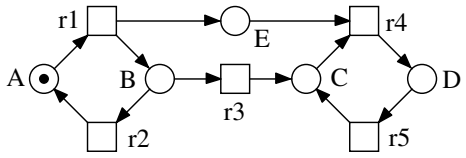
- *every deadlock includes an initially marked trap,*

To optimize computational effort this can be translated into:

- *the maximal trap in every minimal deadlock is initially marked.*

This is only possible if there are no input places. An input place establishes a structural deadlock on its own, in which the maximal trap is empty, and therefore not marked. The DTP can still be decided by structural reasoning only. Its importance becomes clear by the following theorems.

[3] The notion *structural deadlock* has nothing in common with the famous deadlock phenomenon of concurrent processes. The Petri net community has been quite creative in trying to avoid this name clash (co-trap, siphon, tube). However, none of these terms got widely accepted.



structural deadlock:	*trap:*
$\bullet\{A,B\} \subseteq \{A,B\}\bullet$	$\{C,D,E\}\bullet \subseteq \bullet\{C,D,E\}$
pretransitions: $\bullet\{A,B\} = \{r1,r2\}$	posttransitions: $\{C,D,E\}\bullet = \{r4,r5\}$
posttransitions: $\{A,B\}\bullet = \{r1,r2,r3\}$	pretransitions: $\bullet\{C,D,E\} = \{r1,r3,r4,r5\}$

Fig. 11. The token on place A can rotate in the left cycle by repeated firing of r1 and r2. Each round produces an additional token on place E, making this place unbounded. This cycle can be terminated by firing of transition r3, which brings the circulating token from the left to the right side of the Petri net. The place set {A, B} cannot get tokens again as soon as it got clean. Thus, it is a (proper) structural deadlock. On the contrary, the place set {C, D, E} cannot become clean again as soon as it got a token. The repeated firing of r4 and r5 reduces the total token number, but cannot remove all of them. Thus, the place set {C, D, E} is a (proper) trap.

Theorem 1 (Relations between structural and behavioural properties).

1. *A net without structural deadlocks is live.*
2. $ORD \wedge DTP \Rightarrow$ *no dead states*
3. $ORD \wedge ES \wedge DTP \Rightarrow$ *live*
4. $ORD \wedge EFC \wedge DTP \Leftrightarrow$ *live*

The first theorem occasionally helps to decide liveness of unbounded nets. The last theorem is also known as *Commoner's theorem*, published in 1972. Theorems 2-4 have been generalized to non-ordinary nets by requiring homogeneity and non-blocking multiplicity [Sta90]. The proof for ordinary Petri nets can be found in [DE95].

Running example The Deadlock Trap Property holds, but no special net structure class is given, therefore we know now that the net is weakly live, i.e. there is no dead state (DSt). Please note, for our given net we are not able to decide liveness by structural reasoning only.

(5) Dynamic decision of behavioural properties In order to decide liveness and reversibility we need to construct the state space. This could be done according the partial order semantics or the interleaving semantics. To keep things simple in this introductory tutorial we consider here the interleaving semantics only, which brings us to the reachability graph.

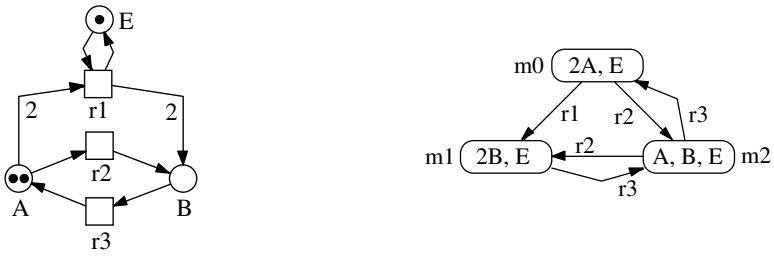


Fig. 12. A Petri net (left) and its reachability graph (right). The states are given in a shorthand notation. In state m0, transitions r1 and r2 are in a dynamic conflict; the firing of one transition disables the other one. In state m2, transitions r2 and r3 are concurrently enabled; they can fire independently, i.e. in any order. In both cases we get a branching node in the reachability graph.

Definition 11 (Reachability graph). *Let $\mathcal{N} = (P, T, f, m_0)$ be a Petri net. The* reachability graph *of $\mathcal{N}$ is the graph $\mathcal{RG}(\mathcal{N}) = (V_\mathcal{N}, E_\mathcal{N})$, where*

- $V_\mathcal{N} := [m_0\rangle$ *is the set of nodes,*
- $E_\mathcal{N} := \{\, (m, t, m') \mid m, m' \in [m_0\rangle, t \in T : m[t\rangle m' \}$ *is the set of arcs.*

The nodes of a reachability graph represent all possible states (markings) of the net. The arcs in between are labelled by single transitions, the firing of which causes the related state change, compare Figure 12. The reachability graph gives us a finite automaton representation of all possible single step firing sequences. Consequently, concurrent behaviour is described by enumerating all interleaving firing sequences; so the reachability graph reflects the behaviour of the net according to the interleaving semantics.

The reachability graph is finite for bounded nets only. A branching node in the reachability graph, i.e. a node with more than one successor, reflects either alternative or concurrent behaviour. The difference is not locally decidable anymore in the reachability graph. For 1-bounded ordinary state machines, net structure and reachability graph are isomorphic.

Reachability graphs tend to be huge. In the worst-case the state space grows faster than any primitive recursive function[4], basically for two reasons: concurrency is resolved by all interleaving sequences, see Figure 13, and the tokens in over-populated P-invariants can distribute themselves fairly arbitrarily, see Figure 14. The state space explosion motivates the static analyses, as discussed in the preceding analysis steps. If we succeed in constructing the complete reachability graph, we are able to decide behavioural Petri net properties.

- A Petri net is k-bounded iff there is no node in the reachability graph with a token number larger than k in any place.

[4] To be precise: the dependence of the size of a reachability graph on the size of the net cannot be bounded by a primitive recursive function [PW03].

- A Petri net is reversible iff the reachability graph is strongly connected.
- A Petri net is deadlock-free iff the reachability graph does not contain terminal nodes, i.e., nodes without outgoing arcs.
- In order to decide liveness, we partition the reachability graph into strongly connected components (SCC), i.e. maximal sets of strongly connected nodes. A SCC is called terminal if no other SCC is reachable in the partitioned graph. A transition is live iff it is included in all terminal SCCs of the partitioned reachability graph. A Petri net is live iff this holds for all transitions.

The occurrence of dynamic conflicts is checked at best during the construction of the reachability graph, because branching nodes do not necessarily mean alternative system behaviour.

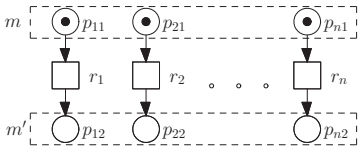


Fig. 13. State explosion problem 1. There are $n!$ interleaving sequences from m (all places p_{*1} carry a token) to m' (all places p_{*2} carry a token), causing $2^n - 2$ intermediate states.

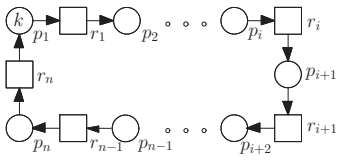


Fig. 14. State explosion problem 2. The k tokens are bound to circulate within the given cycle. They can arbitrarily distribute themselves on the n places, forming a P-invariant. There are $(n + k - 1)!/[(n - 1)!\,k!]$ possibilities for this distribution (combinations with repetition). Each distribution defines a state.

Running example We already know that the net is bounded, so the reachability graph has to be finite. It comprises in the Boolean token interpretation 118 states out of 2^{22} theoretically possible ones; see Table 1 for some samples of the size of the state space in the integer token semantics (discrete concentration levels). Independently of the size, the reachability graph we get forms one strongly connected component. Therefore, the Petri net is reversible, i.e. each system state is always reachable again. Further, each transition (reaction) appears at least once in this strongly connected component, therefore the net is live. There are dynamic conflicts, e.g. between r2 and r3 in all states, where Raf_RasGTP is marked.

Moreover, from the viewpoint of the qualitative model, all of these states of the reachability graph's only strongly connected component are equivalent, and each could be taken as an initial state resulting in exactly the same total (discrete) system behaviour. This prediction will be confirmed by the observations gained during quantitative analyses, see Sections 5.2 and 6.2.

Table 1. State explosion in the running example.

levels	IDD data structure[a] number of nodes	reachability graph number of states
1	52	118
4	115	$2.4 \cdot 10^4$
8	269	$6.1 \cdot 10^6$
40	3,697	$4.7 \cdot 10^{14}$
80	13,472	$5.6 \cdot 10^{18}$
120	29,347	$1.7 \cdot 10^{21}$

[a] This computational experiment has been performed with idd-ctl, a model checker based on interval decision diagrams (IDD).

This concludes the analysis of *general* behavioural net properties, i.e. of properties we can speak about in syntactic terms only, without any semantic knowledge. The next step consists in a closer look at *special* behavioural net properties, reflecting the expected special functionality of the network.

(6) Model checking of special behavioural properties Temporal logic is particularly helpful in expressing special behavioural properties of the expected transient behaviour, whose truth can be determined via model checking. It is an unambiguous language, providing a flexible formalism which considers the validity of propositions in relation to the execution of the model. Model checking generally requires boundedness. If the net is 1-bounded, there exists a particularly rich choice of model checkers, which get their efficiency by exploiting sophisticated data structures and algorithms.

One of the widely used temporal logics is the *Computational Tree Logic* (CTL). It works on the computational tree, which we get by unwinding the reachability graph, compare Figure 15. Thus, CTL represents a branching time logic with interleaving semantics.

The application of this analysis approach requires an understanding of temporal logics. Here, we restrict ourselves to an informal introduction into CTL. CTL - as any temporal logic - is an extension of a classical (propositional) logic. The atomic propositions consist of statements on the current token situation in a given place. In the case of 1-bounded models, places can be read as Boolean variables, with allows propositions such as $RafP$ instead of $m(RafP) = 1$. Likewise, places are read as integer variables for k-bounded models, $k \neq 1$.

Propositions can be combined to composed propositions using the standard logical operators: $\neg$ (negation), $\wedge$ (conjunction), $\vee$ (disjunction), and $\rightarrow$ (implication), e.g. $RafP \wedge ERKP$.

The truth value of a proposition may change by the execution of the net; e.g. the proposition $RafP$ does not hold in the initial state, but there are reachable states where Raf is phosphorylated, so $RafP$ holds in these states. Such temporal relations between propositions are expressed by the additionally available temporal operators.

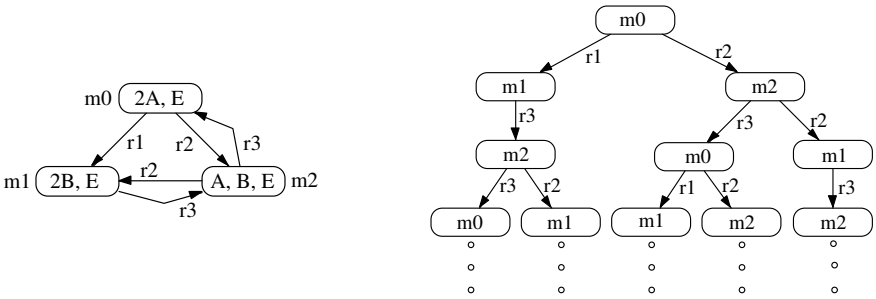


Fig. 15. Unwinding the reachability graph (left) into an infinite computation tree (right). The root of the computation tree is the initial state of the reachability graph.

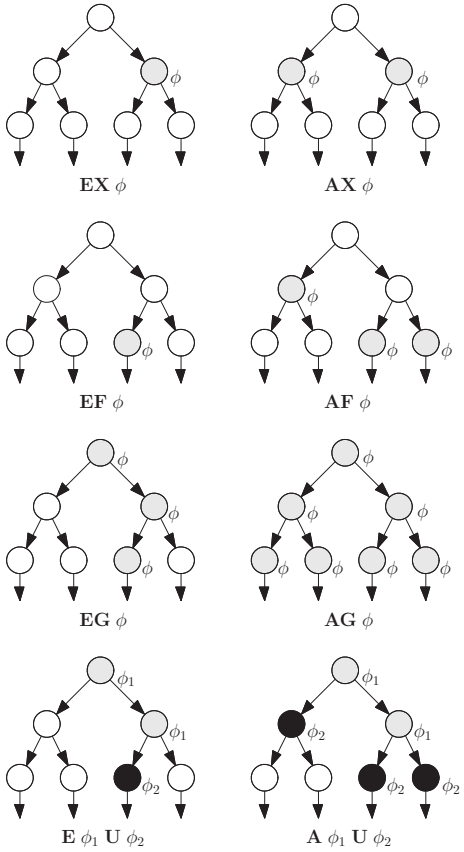


Fig. 16. The eight CTL operators and their semantics in the computation tree, which we get by unwinding the reachability graph, compare Figure 15. The two path quantifiers **E**, **A** relate to the branching structure in the computation tree: **E** - for some computation path (left column), **A** - for all computation paths (right column).

In CTL there are basically four of them (ne**X**t, **F**inally, **G**lobally, **U**ntil), which come in two versions (**E** for Existence, **A** for All), making together eight operators. Let $\phi_{[1,2]}$ be an arbitrary temporal-logic formulae. Then, the following formulae hold in state m,

- **EX** ϕ : if there is a state reachable by one step where ϕ holds.
- **EF** ϕ : if there is a path where ϕ holds finally, i.e., at some point.
- **EG** ϕ : if there is a path where ϕ holds globally, i.e., forever.
- **E** (ϕ_1 **U** ϕ_2) : if there is a path where ϕ_1 holds until ϕ_2 holds.

The other four operators, which we get by replacing the **E**xistence operator by the **A**ll operator, are defined likewise by extending the requirement *"there is a path"* to *"for all paths"*. A formula holds in a net if it holds in its initial state. See Figure 16 for a graphical illustration of the eight temporal operators.

Running example We confine ourselves here to two CTL properties, checking the generalizability of the insights gained by the partial order run of the I/O T-invariant. Recall that places are interpreted as Boolean variables in order to simplify notation.

property Q1: The signal sequence predicted by the partial order run of the I/O T-invariant is the only possible one. In other words, starting at the initial state, it is necessary to pass through states RafP, MEKP, MEKPP and ERKP in order to reach ERKPP.

$$\neg [\ \mathbf{E}\ (\ \neg\ \text{RafP}\ \ \mathbf{U}\ \text{MEKP}\ )\ \vee\ \mathbf{E}\ (\ \neg\ \text{MEKP}\ \ \mathbf{U}\ \text{MEKPP}\ )\ \vee\ \mathbf{E}\ (\ \neg\ \text{MEKPP}\ \ \mathbf{U}\ \ \text{ERKP}\ )\ \vee\ \mathbf{E}\ (\ \neg\ \text{ERKP}\ \ \mathbf{U}\ \ \text{ERKPP}\ )\ \ ]$$

property Q2: Dephosphorylation takes place independently. E.g., the duration of the phosphorylated state of ERK is independent of the duration of the phosphorylated states of MEK and Raf.

$$(\ \mathbf{EF}\ [\ \text{Raf} \wedge (\ \text{ERKP} \vee \text{ERKPP}\ )\ ] \wedge \mathbf{EF}\ [\ \text{RafP} \wedge (\ \text{ERKP} \vee \text{ERKPP}\ )\ ] \wedge \mathbf{EF}\ [\ \text{MEK} \wedge (\ \text{ERKP} \vee \text{ERKPP}\ )\ ] \wedge \mathbf{EF}\ [\ (\ \text{MEKP} \vee \text{MEKPP}\ ) \wedge (\ \text{ERKP} \vee \text{ERKPP}\ )\ ]\ )$$

Temporal logic is an extremely powerful and flexible language to describe special properties, however needs some experience to get accustomed to it. Applying this analysis technique requires seasoned understanding of the network under investigation, combined with the skill to correctly express the expected behaviour in temporal logics.

In subsequent sections we will see how to employ the same technique in a quantitative setting. We will use Q1 as a basis to illustrate how the stochastic and continuous approaches provide complementary views of the system behaviour.

4.3 Summary

To summarize the preceding validation steps, the model has passed the following validation criteria.

- **validation criterion 0** All expected structural properties hold, and all expected general behavioural properties hold.
- **validation criterion 1** The net is CPI, and there are no minimal P-invariant without biological interpretation.
- **validation criterion 2** The net is CTI, and there are no minimal T-invariant without biological interpretation. Most importantly, there is no known biological behaviour without a corresponding, not necessarily minimal, T-invariant.
- **validation criterion 3** All expected special behavioural properties expressed as temporal-logic formulae hold.

One of the benefits of using the qualitative approach is that systems can be modelled and analysed without any quantitative parameters. In doing so, all possible behaviour under any timing is considered. Moreover the qualitative step helps in identifying suitable initial markings and potential quantitative analysis techniques. Now we are ready for a more sophisticated quantitative analysis of our model.

5 The Stochastic Approach

5.1 Stochastic Modelling

As with a qualitative Petri net, a stochastic Petri net maintains a discrete number of tokens on its places. But contrary to the time-free case, a firing rate (waiting time) is associated with each transition t, which are random variables $X_t \in [0, \infty)$, defined by probability distributions. Therefore, all reaction times can theoretically still occur, but the likelihood depends on the probability distribution. Consequently, the system behaviour is described by the same discrete state space, and all the different execution runs of the underlying qualitative Petri net can still take place. This allows the use of the same powerful analysis techniques for stochastic Petri nets as already applied for qualitative Petri nets.

For better understanding we describe the general procedure of a particular simulation run for a stochastic Petri net. Each transition gets its own local timer. When a particular transition becomes enabled, meaning that sufficient tokens arrive on its preplaces, then the local timer is set to an initial value, which is computed at this time point by means of the corresponding probability distribution. In general, this value will be different for each simulation run. The local timer is then decremented at a constant speed, and the transition will fire when the timer reaches zero. If there is more than one enabled transition, a race for the next firing will take place.

Technically, various probability distributions can be chosen to determine the random values for the local timers. Biochemical systems are the prototype for exponentially distributed reactions. Thus, for our purposes, the firing rates of all transitions follow an exponential distribution, which can be described by a single parameter λ, and each transition needs only its particular, generally marking-dependent parameter λ to specify its local time behaviour. The following definition summarises this informal introduction.

Definition 12 (Stochastic Petri net, Syntax). *A* biochemically interpreted stochastic Petri net *is a quintuple* $\mathcal{SPN}_{Bio} = (P, T, f, v, m_0)$*, where*

- P *and* T *are finite, non empty, and disjoint sets.* P *is the set of* places*, and* T *is the set of* transitions*.*
- $f : ((P \times T) \cup (T \times P)) \to \mathbb{N}_0$ *defines the set of directed* arcs*, weighted by nonnegative integer values.*
- $v : T \to H$ *is a function, which assigns a* stochastic hazard function h_t *to each transition* t*, whereby* $H := \bigcup_{t \in T} \left\{ h_t \,|\, h_t : \mathbb{N}_0^{|^\bullet t|} \to \mathbb{R}^+ \right\}$ *is the set of all stochastic hazard functions, and* $v(t) = h_t$ *for all transitions* $t \in T$*.*
- $m_0 : P \to \mathbb{N}_0$ *gives the* initial marking*.*

The stochastic hazard function h_t defines the marking-dependent transition rate $\lambda_t(m)$ for the transition t. The domain of h_t is restricted to the set of preplaces of t to enforce a close relation between network structure and hazard functions. Therefore $\lambda_t(m)$ actually depends only on a sub-marking.

Stochastic Petri net, Semantics Transitions become enabled as usual, i.e. if all preplaces are sufficiently marked. However there is a time, which has to elapse, before an enabled transition $t \in T$ fires. The transition's waiting time is an exponentially distributed random variable X_t with the *probability density function*:

$$f_{X_t}(\tau) = \lambda_t(m) \cdot \mathrm{e}^{(-\lambda_t(m) \cdot \tau)}, \qquad \tau \geq 0.$$

The firing itself does not consume time and again follows the standard firing rule of qualitative Petri nets. The semantics of a stochastic Petri net (with exponentially distributed reaction times for all transitions) is described by a continuous time Markov chain (CTMC). The CTMC of a stochastic Petri net without parallel transitions is isomorphic to the reachability graph of the underlying qualitative Petri net, while the arcs between the states are now labelled by the transition rates. For more details see [MBC+95], [BK02].

Based on this general $\mathcal{SPN}_{Bio}$ definition, specialised biochemically interpreted stochastic Petri nets can be defined by specifying the required kind of stochastic hazard function more precisely. We give two examples, reading the tokens as molecules or as concentration levels. The *stochastic mass-action hazard function* tailors the general $\mathcal{SPN}_{Bio}$ definition to biochemical mass-action networks, where tokens correspond to molecules:

$$h_t := c_t \cdot \prod_{p \in {}^\bullet t} \binom{m(p)}{f(p,t)}, \tag{1}$$

where c_t is the transition-specific stochastic rate constant, and $m(p)$ is the current number of tokens on the preplace p of transition t. The binomial coefficient

describes the number of unordered combinations of the $f(p,t)$ molecules, required for the reaction, out of the $m(p)$ available ones.

Tokens can also be read as concentration levels, as introduced in [CVGO06]. The current concentration of each species is given as an abstract level. We assume the maximum molar concentration is M, and the amount of different levels is $N+1$. Then the abstract values $0,\ldots,N$ represent the concentration intervals $0,\ (0,\ 1*M/N],\ (1*M/N,\ 2*M/N],\ \ldots,\ (N-1*M/N,\ N*M/N]$. Each of these (finitely many) discrete levels stands for an equivalence class of (infinitely many) continuous states. The *stochastic level hazard function* tailors the general $\mathcal{SPN}_{Bio}$ definition to biochemical mass-action networks, where tokens correspond to concentration levels; for ordinary nets we get:

$$h_t := k_t \cdot N \cdot \prod_{p \in {}^\bullet t} \left(\frac{m(p)}{N}\right), \tag{2}$$

where k_t is the transition-specific deterministic rate constant, and N the number of the highest level. The transformation rules between the stochastic and deterministic rate constants are well-understood, see e.g. [Wil06]. In practice, kinetic rates are taken from literature, textbooks, etc. or determined from biochemical experiments. A hazard function (2) is the means whereby the continuous model (see the framework in Figure 2 and Section 6) can be approximated by the stochastic model; this can generally be achieved by a limited number of levels – see Section 5.2.

We only consider here the level semantics. Since the continuous concentrations of proteins in our running example are all in the same range (0.1...0.4 mMol in 0.1 steps), we employ a model with only 4, and a second version with 8 levels, compare Figure 17.
The corresponding CTMCs (and reachability graphs) comprise 24,065 states for the 4 level version and 6,110,643 states for the 8 level version, compare Table 1.

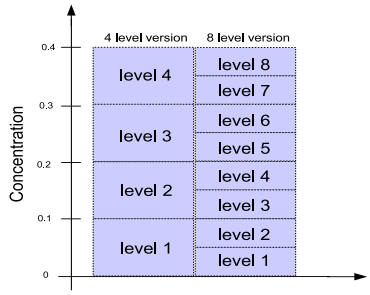


Fig. 17. The partitioning of the concentration scale into discrete levels.

5.2 Stochastic Analysis

Due to the isomorphy of the reachability graph and the CTMC, all qualitative analysis results obtained in Section 4 are still valid. The influence of time does not restrict the possible system behaviour. Specifically it holds that the CTMC of our case study is reversible, which ensures ergodicity; i.e. we could start the

system in any of the reachable states, always resulting in the same CTMC with the same steady state probability distribution.

Additionally, probabilistic analyses of the transient and steady state behaviour are now available. Generally, this can be done in an analytical as well as in a simulative manner. The analytical approach works on the CTMC, which therefore has to be finite. Consequently, the net to be analysed has to be bounded. On the contrary, the simulative approach works also for systems with infinite state space or state space beyond the current limits of exact analyses, and for systems with complex dynamics as semi-Markov processes or generalized semi-Markov processes.

In order to use the probabilistic model checker PRISM [PNK06] for the analytical approach, we encode the running example in its modelling language. We follow the technique proposed in [DDS04], which is more natural for Petri nets than the one proposed in [CVGO06] for algebraic models. This translation requires knowledge of the boundedness degree of all species involved, which we acquire by the structural analysis technique of P-invariants.

In the following the reader is assumed to be familiar with related standard techniques and terminology.

(1) Equivalence check by transient analysis We start with transient analysis to prove the sufficient equivalence between the stochastic model in the level semantics and the corresponding continuous model, justifying the interpretation of the properties gained by the stochastic model also in terms of the continuous one. PRISM permits the analysis of the transient behaviour of the stochastic model; e.g., the concentration of RafP at time t is given by:

$$\mathrm{C}_{RafP}(t) \;=\; \tfrac{0.1}{s} \cdot \underbrace{\sum_{i=1}^{4s} \big(i \cdot P(L_{RafP}(t) = i)\big)}_{expected\ value\ of\ L_{RafP}(t)} \;.$$

The random variable $L_{RafP}(t)$ stands for the level of RafP at time t. We set s to 1 for the 4 level version, and to 2 for the 8 level version. The factor $\frac{0.1}{s}$ calibrates the expected value for a given level to the concentration scale. In the 4 level version a single level (token) represents 0.1 mMol and 0.05 mMol in the 8 level version. Figure 18 shows the simulation results for the species MEK and RasGTP in the time interval [0..100] according to the continuous and the stochastic models respectively. These results confirm that 4 levels are sufficiently adequate to approximate the continuous model, and that 8 levels are preferable if the computational expenses are acceptable.

(2) Analytical stochastic model checking In Section 4.2 we employed CTL to express behavioural properties. Since we have now a stochastic model, we apply Continuous Stochastic Logic (CSL), which replaces the path quantifiers (**E**, **A**) in CTL by the probability operator $\mathbf{P}_{\bowtie p}$, whereby $\bowtie p$ specifies the probability of the given formula. For example, introducing in CSL the abbreviation

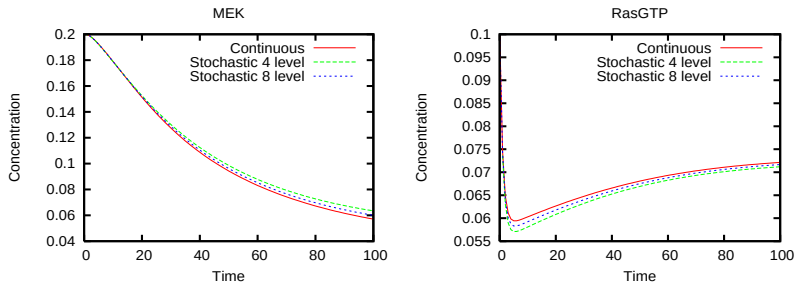


Fig. 18. Comparison of the concentration traces.

$\mathbf{F}\,\phi$ for $\mathit{true}\,\mathbf{U}\,\phi$, the CTL formula $\mathbf{EF}\,\phi$ becomes the CSL formula $\mathbf{P}_{\geq 0}[\,\mathbf{F}\,\phi\,]$, and $\mathbf{AF}\,\phi$ becomes $\mathbf{P}_{\geq 1}[\,\mathbf{F}\,\phi\,]$.

We give two properties related to the partial order run of the I/O T-invariant, see Section 4.2 and qualitative property Q1 therein, from which we expect a consecutive increase of RafP, MEKPP and ERKPP. Both properties are expressed as so-called experiments, which are analysed varying the parameter L over all levels, i.e. 0 to N. For the sake of efficiency, we restrict the **U** operator to 100 time steps. Note that places are read as integer variables in the following.

property S1a: What is the probability of the concentration of RafP increasing, when starting in a state where the level is for the first time at L (the latter side condition is specified by the filter given in braces)?

$$\mathbf{P}_{=?}\,[\,(\,\mathrm{RafP}\;=\;\mathrm{L}\,)\;\mathbf{U}^{<=100}\;(\,\mathrm{RafP} > \mathrm{L}\,)\,\{\,\mathrm{RafP} = \mathrm{L}\,\}\,]$$

The results indicate, see Figure 19(a), that it is absolutely certain that the concentration of RafP increases from level 0 and likewise there is no increase from level N; this behaviour has already been determined by the qualitative analysis. Furthermore, an increase in RafP is very likely in the lower levels, increase and decrease are almost equally likely in the intermediate levels, while in the higher levels, but obviously not in the highest, an increase is rather unlikely (but not impossible). In summary this means that the total mass, circulating within the first layer of the signalling cascade, is unlikely to be accumulated in the activated form. We need this understanding to interpret the results for the next property.

property S2a: What is the probability that, given the initial concentrations of RafP, MEKPP and ERKPP being zero, the concentration of RafP rises above some level L while the concentrations of MEKPP and ERKPP remain at zero, i.e. RafP is the first species to react?

$$\begin{aligned}\mathbf{P}_{=?}\,[\,(\,(\,\mathrm{MEKPP}\;=\;0\,)\;\wedge\;(\,\mathrm{ERKPP}\;=\;0\,)\,)\;\mathbf{U}^{<=100}\;(\,\mathrm{RafP} >\;\mathrm{L}\,)\\ \{\,(\,\mathrm{MEKPP}\;=\;0\,)\;\wedge\;(\,\mathrm{ERKPP}\;=\;0\,)\;\;\wedge\;(\,\mathrm{RafP} =\;0\,)\,\}\;]\end{aligned}$$

The results indicate, see Figure 19(b), that the likelihood of the concentration of RafP rising, while those of MEKPP and ERKPP are zero, is very high in

the bottom half of the levels, and quite high in the lower levels of the upper half. The decrease of the likelihood in the higher levels is explained by property S1. Property S2 is related to the qualitative property Q1 (Section 4.2), and the continuous property C1 (Section 6.2) – the concentration of RafP rises before those of MEKPP and ERKPP.

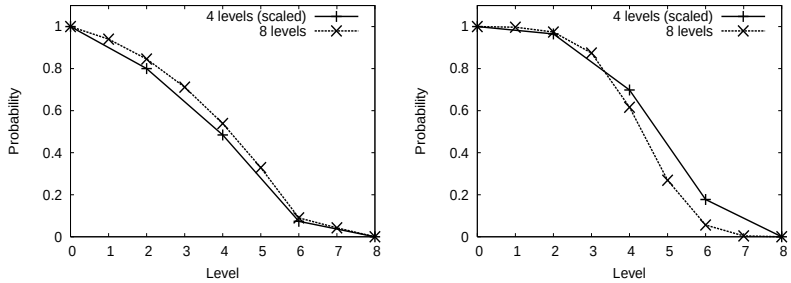


Fig. 19. Probability of the accumulation of RafP. (a) property S1. (b) property S2.

Due to the computational efforts of analytical stochastic model checking, we are only able to treat properties over a stochastic model with 4 or at most 8 levels. This restricts the kind of properties that we can prove; e.g., in order to check increases of MEKPP and ERKPP – as suggested by the qualitative property Q1 and done above for RafP in the stochastic properties S1 and S2 – we would need 50 or 200 levels respectively.

Analytical stochastic model checking becomes more and more impractical with increasing size of the state space. In order to avoid the enormous computational power required for larger state spaces, the time-dependent stochastic behaviour can be simulated by dedicated algorithms, and evaluated by simulative stochastic model checking, see next step, or approximated by a deterministic continuous behaviour, see Section 6.

(3) Simulative stochastic model checking This approach of Monte Carlo sampling handles large state spaces through approximating results by analysing only a subset of the state space – a set of finite outputs from a stochastic simulation algorithm (SSA), e.g. Gillespie's exact SSA [Gil77].

The type of logic now suitable for describing properties changes from branching time (e.g., CSL operating over CTMC) to linear time. A linear time logic operates in-turn over sets of linear paths through the state space, equivalent to operating on simulation outputs. A given property holds if it holds in all paths. Consequently, there are no path quantifiers in LTL.

We apply PLTL, a probabilistic linear time temporal logic [MC208]. This logic extends standard LTL to a stochastic setting, with a $\mathbf{P}_{\bowtie p}$ operator, such as in CSL, and a filter construct, $\{\,\phi\,\}$, defining the initial state of the property.

However, PLTL does not have the ability to embed probability operators or perform steady state analysis.

The semantics is defined over sets of linear traces of temporal behaviour, in this case by stochastic simulation runs. Each trace is evaluated to a Boolean truth value, and the probability of a property holding true is computed by the fraction of true values in the set over the whole set. Please note, the choice of simulator and simulation parameters used to compute the sequence of states can affect the semantics of the PLTL property and the correctness of the result.

This approach to model checking incorporates two approximations. The truth value of a single trace is approximated by operating over a finite sequence of states only; and the probability of the property is approximated through sampling a finite number of traces (a subset of the model's behaviours) only.

PLTL could be considered as a linear time counterpart to CSL, and can easily be used to formalise the visual evaluation of diagrams as generated by deterministic/stochastic simulation runs or by recording experimental time series. We repeat properties S1a and S2a. Notice that the properties no longer require time bounds on temporal operators.

property S1s: What is the probability of the concentration of RafP increasing, when starting in a state where the level is for the first time at L (the latter side condition is specified by the filter given in braces)?

$$\mathbf{P}_{=?}\ [\ (\ \mathrm{RafP}\ =\ \mathrm{L}\ )\ \mathbf{U}\ (\ \mathrm{RafP} > \mathrm{L}\ )\{\ \mathrm{RafP} = \mathrm{L}\ \}\ ]$$

We check this property using 100 simulation traces from Gillespie's algorithm with a simulation time of 300s as input to the PLTL model checker MC2 [MC208].

This property can be assessed with far greater numbers of tokens than possible in the analytical approach. We highlight the efficiency of the simulative approach in Table 5.2 providing the time taken at varying numbers of tokens to perform model checking.

We extend the analysis of property S1 up to 4,000 molecules, shown in Figure 20, and observe that when increasing the number of molecules, the behaviour of the pathway tends towards the deterministic behaviour. The deterministic behaviour states that the protein RafP will always increase (property probability 1) until it reaches its maximum concentration value of around 0.1182 $mMol$. With increasing molecules, the maximum possible number of molecules in the stochastic behaviour of RafP tends towards the deterministic maximum (vertical line). The stochastic behaviour is seen to tend towards a probability of 0.5 in its possible concentration range, due to the stochastic nature where there is always a possibility of the protein decreasing or increasing when at a certain concentration.

property S2s: What is the probability that, given the initial concentrations of RafP, MEKPP and ERKPP being zero, the concentration of RafP rises above some level L while the concentrations of MEKPP and ERKPP remain at zero, i.e. RafP is the first species to react?

Table 2. Example figures for MC2 model checking of property S1 at varying number of levels/molecules.

Levels	MC Time[a]	Simulation Output Size
4	10 s [b]	750 KB
8	15 s [b]	1.5 MB
40	1.5 minutes [b]	7.5 MB
400	1 minute [c]	80 MB
4,000	30 minutes [c]	900 MB

[a] Both Gillespie simulation and MC2 checking.
[b] Computation on a standard workstation.
[c] Distributed computation on a computer cluster comprising 45 Sun X2200 servers each with 2 dual core processors (180 CPU cores).

$$\mathbf{P}_{=?}\ [\ (\ (\ \mathrm{MEKPP}\ =\ 0\ )\ \wedge\ (\ \mathrm{ERKPP}\ =\ 0\ )\ )\ \mathbf{U}\ (\ \mathrm{RafP} >\ \mathrm{L}\ )$$
$$\{\ (\ \mathrm{MEKPP}\ =\ 0\ )\ \wedge\ (\ \mathrm{ERKPP}\ =\ 0\ )\ \ \wedge\ (\ \mathrm{RafP} =\ 0\ )\ \}\ ]$$

To perform the analysis, we use the same simulation time (300s) and number of runs (100) as per S1s. Similarly, we extend this analysis up to 4,000 molecules, shown in Figure 21, and again note that the stochastic behaviour begins to approximate the deterministic behaviour. In the deterministic behaviour, only at the initial state of the system are RafP, MEKPP and ERKPP all zero, hence a probability of 1 at this state and probability of 0 elsewhere. With increasing molecules, the stochastic behaviour becomes less curved and more step-like, tending towards the vertical line in the deterministic behaviour.

5.3 Summary

The stochastic Petri net contains discrete tokens and transitions which fire probabilistically. In summary, our results show that

1. Transient analysis helps to decide on the number of tokens to adequately describe the system.
2. The stochastic behaviour tends towards the deterministic behaviour. Thus the stochastic model can be approximated by a continuous model, representing the averaged behaviour only.
3. Stochastic model checking allows a quantification of the probabilities at which qualitative properties hold.

6 The Continuous Approach

6.1 Continuous Modelling

In a continuous Petri net the marking of a place is no longer an integer, but a positive real number, called token value, which we are going to interpret as the concentration of the species modelled by the place. The instantaneous firing of a transition is carried out like a continuous flow.

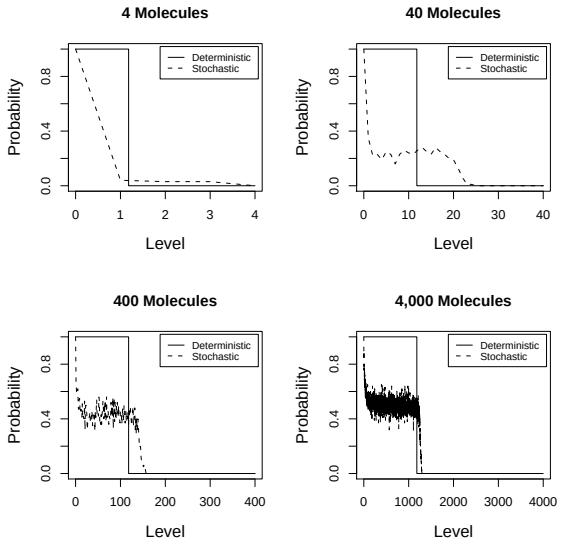


Fig. 20. Simulative stochastic model checking for property S1 at a varying number of molecules; 4, 40, 400 and 4,000. This shows a progression towards the deterministic behaviour as the number of molecules increases.

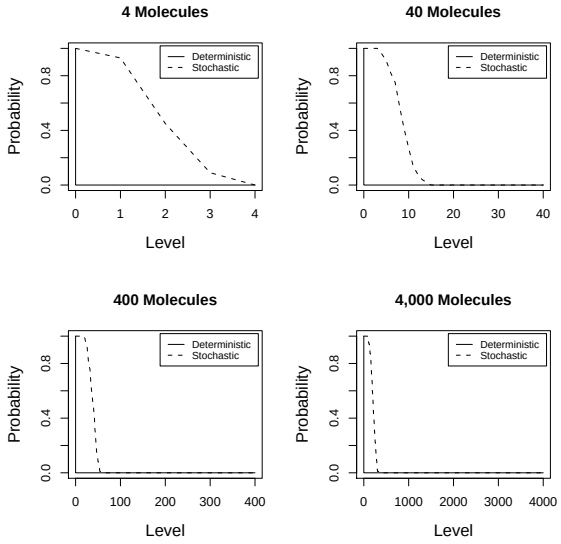


Fig. 21. Simulative stochastic model checking for property S2 at a varying number of molecules; 4, 40, 400 and 4,000. This shows a progression towards the deterministic behaviour as the number of molecules increases.

Definition 13 (Continuous Petri net, Syntax). *A* continuous Petri net *is a quintuple* $\mathcal{CON}_{Bio} = (P, T, f, v, m_0)$*, where*

- P *and* T *are finite, non empty, and disjoint sets.* P *is the set of* continuous places. T *is the set of* continuous transitions.
- $f : ((P \times T) \cup (T \times P)) \to \mathbb{R}_0^+$ *defines the set of directed* arcs*, weighted by nonnegative real values.*
- $v : T \to H$ *is a function which assigns a* firing rate function h_t *to each transition* t*, whereby*
  $H := \bigcup_{t \in T} \left\{ h_t | h_t : \mathbb{R}^{|{}^\bullet t|} \to \mathbb{R} \right\}$ *is the set of all firing rate functions, and* $v(t) = h_t$ *for all transitions* $t \in T$*.*
- $m_0 : P \to \mathbb{R}_0^+$ *gives the* initial marking.

The firing rate function h_t defines the marking-dependent continuous transition rate for the transition t. The domain of h_t is restricted to the set of preplaces of t to enforce a close relation between network structure and firing rate functions. Therefore $h_t(m)$ actually depends only on a sub-marking.

Technically, any mathematical function in compliance with this restriction is allowed for h_t. However, often special kinetic patterns are applied, whereby Michaelis-Menten and mass-action kinetics seem to be the most popular ones.

Please note, a firing rate may also be negative, in which case the reaction takes place in the reverse direction. This feature is commonly used to model reversible reactions by just one transition, where positive firing rates correspond to the forward direction, and negative ones to the backward direction.

Continuous Petri net, Semantics Each continuous marking is a place vector $m \in \left(\mathbb{R}_0^+\right)^{|P|}$, and $m(p)$ yields again the marking on place p, which is now a real number. A continuous transition t is enabled in m, if $\forall p \in {}^\bullet t : m(p) > 0$. Due to the influence of time, a continuous transition is forced to fire as soon as possible.

The semantics of a continuous Petri net is defined by a system of ODEs, whereby one equation describes the continuous change over time on the token value of a given place by the continuous increase of its pretransitions' flow and the continuous decrease of its posttransitions' flow, i.e., each place p subject to changes gets its own equation:

$$\frac{dm\,(p)}{dt} = \sum_{t \in {}^\bullet p} f\,(t,p)\,v\,(t) - \sum_{t \in p^\bullet} f\,(p,t)\,v\,(t)\,,$$

Each equation corresponds basically to a line in the incidence matrix, whereby now the matrix elements consist of the rate functions multiplied by the arc weight, if any.

In other words, the continuous Petri net becomes the structured description of the corresponding ODEs. Due to the explicit structure we expect to get descriptions which are less error prone compared to those ones created manually

from the scratch. In fact, writing down a system of ODEs by designing continuous Petri nets instead of just using a text editor might be compared to high-level instead of assembler programming.

For our running case study, we derive the continuous model from the qualitative Petri net by associating a *mass action rate* with each transition in the network.

We can likewise derive the continuous Petri net from the stochastic Petri net by approximating over the hazard function of type (1), see for instance [Wil06]. In both cases, we obtain a *continuous Petri net*, preserving the structure of the qualitative one, see our framework in Figure 2.

The complete system of nonlinear ODEs generated from the continuous Petri net of our running example is given in [GHL07a], Appendix C.

The initial concentrations as suggested by the qualitative analysis correspond to those given in [LBS00], when mapping nonzero values to 1. For reasons of better comparability we have also considered more precise initial concentrations, where the presence of a species is encoded by biologically motivated real values varying between 0.1 and 0.4 in steps of 0.1.

6.2 Continuous Analysis

As soon as there are transitions with more than one preplace, we get a non-linear system, which calls for a numerical treatment of the system on hand. In order to simulate the continuous Petri net, exactly the same algorithms are employed as for numerical differential equation solvers.

In the following the reader is assumed to be familiar with related standard techniques and terminology.

(1) Steady state analysis Since there are 22 species, there are 2^{22}, i.e. 4,194,304 possible initial states in the qualitative Petri net (Boolean token interpretation). Of these, 118 were identified by the reachability graph analysis (Section 4.2) to form one strongly connected component, and thus to be "good" initial states. These are 'sensible' initial states from the point of view of biochemistry in that in all these 118 cases, and in none of the other 4,194,186 states, each protein species is in a high initial concentration in only one of the following states: uncomplexed, complexed, unphosphorylated or phosphorylated. These conditions relate exactly to the 1-P-invariant interpretation given in our initial marking construction procedure in Section 4.2.

We then compute the steady state of the set of species for each possible initial state, using the MatLab ODE solver ode45, which is based on an explicit Runge-Kutta formula, the Dormand-Prince pair [DP80], with 350 time steps.

In Figure 22 (a) we reproduce the computed behaviour of MEK for all 118 good initial states, showing that despite differences in the concentrations at early time points, the steady state concentration is the same in all 118 states. We also reproduce two arbitrary chosen simulations of the model, compare Figure 23. The

equivalence of the final states, compared with the difference in some intermediate states is clearly illustrated in these figures.

In summary, our results show that all of the 'good' 118 states result in the same set of steady state values for the 22 species in the pathway, within the bounds of computational error of the ODE solver. In [GHL07b] it is also shown that none of the remaining possible initial states results in a steady state close to that generated by the 118 markings in the reachability graph. See Table 4 in [GHL07a], Appendix C for the steady state concentrations of the 22 species.

This is an interesting result, because the net considered here is not covered by the class of net structures discussed in [ADLS06] with the unique steady state property.

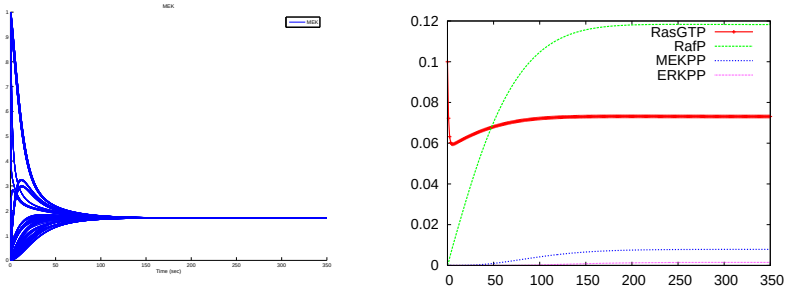


Fig. 22. (a) Steady state analysis of MEK for all 118 'good' states. (b) Continuous transient analysis of the phosphorylated species RasP, MEKPP, ERKPP, triggered by RasGTP.

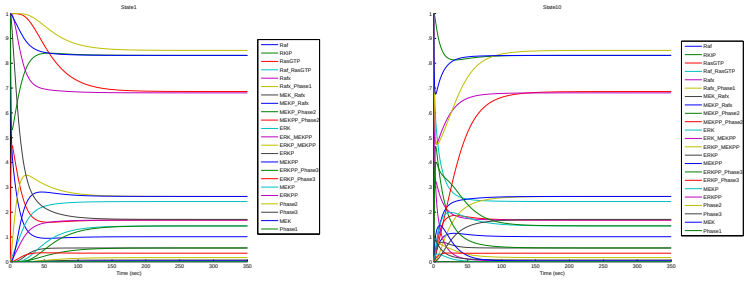


Fig. 23. Dynamic behaviour for state 1 (left) and state 10 (right). State 1 corresponds to the initial marking suggested by Section 4.2.

(2) Continuous model checking of the transient behaviour Corresponding to the partial order run of the I/O T-invariant, see Section 4.2, we expect a consecutive increase of RafP, MEKPP, ERKPP, which we get confirmed by the transient behaviour analysis, compare Figure 22 (b). To formalise the visual evaluation of the diagram we use the continuous linear logic LTLc [CCRFS06] and PLTL [MC208] in a deterministic setting. Both are interpreted over the continuous simulation trace of ODEs.

The following three queries confirm together the claim of the expected propagation sequence. In the queries we have to refer to absolute values. The steady state values are obtained from the steady state analysis in the previous section; these are 0.12 mMol for RafP, 0.008 mMol for MEKPP and 0.002 mMol for ERKPP, all of them being zero in the initial state. If a species' concentration is above half of its steady state value, we call this concentration level significant. Note that in order to simplify the notation, places are interpreted as real variables in the following.

property C1: The concentration of RafP rises to a significant level, while the concentrations of MEKPP and ERKPP remain close to zero; i.e. RafP is really the first species to react.

$$((\text{MEKPP} < 0.001) \wedge (\text{ERKPP} < 0.0002)) \mathbf{U} (\text{RafP} > 0.06)$$

property C2: if the concentration of RafP is at a significant concentration level and that of ERKPP is close to zero, then both species remain in these states until the concentration of MEKPP becomes significant; i.e. MEKPP is the second species to react.

$$((\text{RafP} > 0.06) \wedge (\text{ERKPP} < 0.0002)) \Rightarrow ((\text{RafP} > 0.06) \wedge (\text{ERKPP} < 0.0002)) \mathbf{U} (\text{MEKPP} > 0.004)$$

property C3: if the concentrations of RafP and MEKPP are significant, they remain so, until the concentration of ERKPP becomes significant; i.e. ERKPP is the third species to react.

$$((\text{RafP} > 0.06) \wedge (\text{MEKPP} > 0.004)) \Rightarrow ((\text{RafP} > 0.06) \wedge (\text{MEKPP} > 0.004)) \mathbf{U} (\text{ERKPP} > 0.0005)$$

Note that properties C1, C2 and C3 correspond to the qualitative property Q1, and that S2 is the stochastic counterpart of C1.

We recast these three continuous properties to PLTL and perform model checking using MC2, which is fed with deterministic simulation traces up to simulation time 400s, produced with the BioNessie simulator. A comparison of the MC2 results to the Biocham results is summarised in Table 3.

The difference in the results is due to the different ODE solvers used in BioNessie and Biocham. Due to the adaptive time steps used in Biocham's ODE solver, no state information is outputted for an important time period which is a counter-example to the C2 query, shown in Figure 24. The fixed time step and sufficient granularity of time points used in BioNessie do provide state informa-

Table 3. The results for the replication of C1, C2 and C3 queries in MC2, showing a discrepancy in C2 with the Biocham results.

Query	Biocham	BioNessie & MC2
C1	**true**	**true**
C2	**true**	**false**
C3	**true**	**true**

tion which is a counter-example to this query, thus resulting in a false value. This is an example of where the simulator choice affects the model checking result.

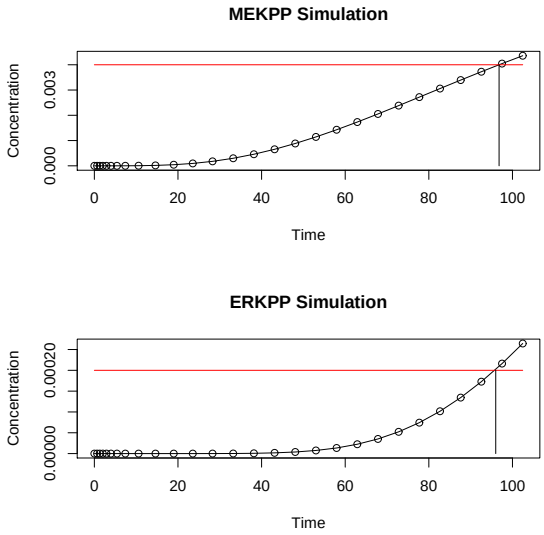


Fig. 24. The output of Biocham simulation showing that it does not output a state in the time period where ERKPP (bottom) > 0.0002 before MEKPP (top) > 0.004, which is a counter example to C2.

6.3 Summary

The continuous Petri net contains tokens with a continuous value and continuous firing of transitions. In summary, our results show that

1. All of the 118 good states identified by the reachability graph of the validated qualitative Petri net result in the same set of steady state values for the 22 species in the pathway.

2. None of the remaining possible initial states of the qualitative Petri net in the Boolean semantics results in a final steady state close to that generated by the good initial markings in the reachability graph.
3. Model checking this Petri net, which contains only a single deterministic behaviour, is akin to analysing the average behaviour of the system. We have shown that the properties as derived from the partial order run of the qualitative model also hold in the average behaviour.

7 Tools

The running example in its interpretation as the three Petri net models have been done using Snoopy [Sno08], a tool to design and animate or simulate hierarchical graphs, among them the qualitative, stochastic and continuous Petri nets as used in this chapter. Snoopy provides export to various analysis tools as well as Systems Biology Markup Language (SBML) [HFS+03] import and export [HRS08].

The qualitative analyses have been made with the Petri net analysis tool Charlie [Cha08]. Charlie's result vector is inspired by the Integrated Net Analyser INA [SR99], the analysis tool we have used for about 20 years. The exploration of the state space growth for increasing level numbers has been done with idd-ctl, a CTL model checker and reachability analyser utilising interval decision diagrams for concise state space representations [Tov06]. The Model Checking Kit [SSE03] has been used for qualitative model checking of 1-bounded models.

The quantitative analyses have been done using Snoopy's build-in simulation algorithms for stochastic and continuous Petri nets, and by BioNessie [Bio08], an SBML-based simulation and analysis tool for biochemical networks. Additionally, MATLAB [SR97] was used to produce the steady state analysis of all initial states in the continuous model.

We employed PRISM [PNK06] for probabilistic model checking of branching time logic, MC2 [MC208], a model checker by Monte Carlo sampling, for probabilistic and continuous model checking of linear time logic, and Biocham [CCRFS06] for LTLc-based continuous model checking.

More Petri nets tools and related material can be found on the Petri Nets World's web page: http://www.informatik.uni-hamburg.de/TGI/PetriNets/.

8 Further Reading and Related Work

Petri nets, as we understand them today, have been initiated by concepts proposed by Carl Adam Petri in his Ph.D. thesis in 1962 [Pet62]. The first substantial results making up the still growing body of Petri net theory appeared around 1970. Initial textbooks devoted to Petri nets were issued in the beginning of the eighties. General introductions into Petri net theory can be found, for example, in [Mur89], [Rei82], [Sta90]; for a comprehensive textbook covering extended free choice nets see specifically [DE95]. An excellent textbook for theoretical issues

is [PW03], which however is in German. The text [DJ01] might be useful, if you just want to get the general flavour in reasonable time.

Petri nets have been employed for technical and administrative systems in numerous application domains since the mid-seventies. The employment in systems biology has been first published in [Hof94], [RML93]. Recent surveys on applying Petri nets for biochemical networks are [Cha07] and [MLM06], offering a rich choice of further reading pointers, among them numerous case studies applying Petri nets to biochemical networks. Besides the net classes introduced in this chapter, coloured Petri nets, duration and interval time Petri nets as well as hybrid Petri nets in various extensions have been employed.

Stochastic Petri nets are an established concept for performance and dependability analysis of technical systems, see [MBC+95], [BK02], recently extended by probabilistic model checking [DDS04]. An excellent textbook for numerical solution of Markov chains is [Ste94]. An overview on stochastic issues for systems biology is given in [Wil06]. The approximation of continuous behaviour by the discretisation of species' concentrations by a finite number of levels has been proposed in [CVGO06]. The application of stochastic Petri nets to biochemical networks was first proposed in [GP98], where they were applied to a gene regulatory network. Further case studies are discussed in [SPB01], [MSS05], [ST05], [SSW05], [Cur06]. A precise definition of biochemically interpreted stochastic Petri nets has been introduced in [GHL07b].

A comprehensive survey on timed Petri net concepts, among them continuous and hybrid Petri nets, however not stochastic Petri nets, in the context of technical systems can be found in [DA05]. See [MFD+03] for cases studies employing hybrid Petri nets to model and analyse biochemical pathways. A precise definition of biochemically interpreted continuous Petri nets has been introduced in [GH06].

P- and T-invariants are well-known concepts of Petri net theory since the very beginning [Lau73]. There are corresponding notions in systems biology, called chemical moieties, elementary modes and extreme pathways, which are elaborated in the setting of biochemical networks in [Pal06]. For biochemical systems without reversible reactions, the notions T-invariants, elementary modes and extreme pathways coincide. The validation of biochemical networks by means of T-invariants is demonstrated in [HK04].

Model checking has been very popular for the verification of technical systems since the eighties. A good starting point for qualitative model checking (CTL, LTL) is [CGP01]. For biochemical networks, qualitative model checking (Boolean semantics) has been introduced in [EKL+02], [CF03], and analytical stochastic model checking in [CVGO06], [HKN+06]. CSL, the stochastic counterpart to CTL, has been originally introduced in [ASSB00], and extended in [BHHK03]. PRISM [PNK06] provides an efficient numerical implementation for CSL model checking, also exploiting symbolic representations. Approximative model checking of CSL using discrete event simulation of probabilistic models has been proposed in [YS02] and implemented in the tool Ymer [YKNP06]. The simulative stochastic model checker MC2 has been inspired by the idea of approx-

imative LTL checking of deterministic simulation runs, proposed in [APUM03], [CCRFS06], [FR07].

The Biocham approach [CCRFS06], as it stands now, restricts itself to a branching time Boolean semantics for qualitative models, while we consider the more general case of integer semantics, which may collapse to the Boolean one. Petri net based model checking also supports the partial order semantics. To analyse continuous models, Biocham provides LTLc, which we used for continuous model checking of the transient behaviour. Meanwhile, this step is facilitated substantially by the extension introduced in [FR07], which permits the inference of the variable values fulfilling a given temporal property from a (set of) continuous simulation run(s).

Finally, there is a lot of activity relating stochastic and continuous models – see [TSB04] for a review. Most work has ignored analysis but instead focussed on simulation, at the molecular, inherently stochastic, level and the population (continuous) level using differential equations, possibly stochastic, e.g. [AME04], [Kie02] and [SK05].

The relation between systems of ordinary differential equations and the net structure of the underlying Petri nets are discussed in [ADLS06].

The systematic qualitative analysis of a metabolic network is demonstrated in [KH08], following basically the same outline as used in section 4.2. How to combine qualitative and quantitative analysis techniques is elaborated in [HDG08] for another signal transduction network and in [GHR$^+$08] for a gene transduction network.

9 Summary

In this paper we have described an overall framework that relates the three major ways of modelling biochemical networks – qualitative, stochastic and continuous – and illustrated this in the context of Petri nets. In doing so we have given a precise definition of biochemically interpreted stochastic and continuous Petri nets. We have shown that the qualitative time-free description is the most basic, with discrete values representing numbers of molecules or levels of concentrations. The qualitative description abstracts over two timed, quantitative models. In the stochastic description, discrete values for the amounts of species are retained, but a stochastic rate is associated with each reaction. The continuous model describes amounts of species using continuous values and associates a deterministic rate with each reaction. These two time-dependent models can be mutually approximated by hazard functions belonging to the stochastic world.

We have illustrated our framework by considering qualitative, stochastic and continuous Petri net descriptions of the ERK signalling pathway, based on the model from [LBS00]. We have focussed on analysis techniques available in each of these three paradigms, in order to illustrate their complementarity. Timing diagrams as produced by numerical simulation techniques are much harder to assess in term of plausibility. That is why we start with qualitative analyses to increase our confidence in the model structure. Our special emphasis has been on

model checking, which is especially useful for transient behaviour analysis, and we have demonstrated this by discussing related properties in the qualitative, stochastic and continuous paradigms. Although our framework is based on Petri nets, it can be applied more widely to other formalisms which are used to model and analyse biochemical networks.

The models developed over the three paradigms share the same structure, so they should share some properties too. However, the interrelationships between these models are not properly understood, yet.

Acknowledgements The running case study has been partly carried out by Sebastian Lehrack during his study stay at the Bioinformatics Research Centre of the University of Glasgow. This stay was supported by the Max Gruenebaum Foundation [MGF] and the UK Department of Trade and Industry Beacon Bioscience Programme.

We would like to thank Rainer Breitling, Richard Orton and Xu Gu for the constructive discussions as well as Vladislav Vyshermirsky for his support in the computational experiments.

References

[ADLS06] D. Angeli, P. De Leenheer, and E.D. Sontag. On the structural monotonicity of chemical reaction networks. In *Proc. 45th IEEE Conference on Decision and Control*, pages 7–12, 2006.

[AME04] D. Adalsteinsson, D. McMillen, and T.C. Elston. Biochemical network stochastic simulator (BioNetS): software for stochastic modeling of biochemical networks. *BMC Bioinformatics*, 5:24, 2004.

[APUM03] M. Antoniotti, A. Policriti, N. Ugel, and B. Mishra. Model building and model checking for biochemical processes. *Cell Biochemistry and Biophysics*, 38:271–286, 2003.

[ASSB00] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model checking continuous time Markov chains. *ACM Trans. on Computational Logic*, 1(1):162–170, 2000.

[BHHK03] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model checking algorithms for continuous time Markov chains. *IEEE Trans. on Software Engineering*, 29(6):524–541, 2003.

[Bio08] BioNessie website. A biochemical pathway simulation and analysis tool. University of Glasgow, http://www.bionessie.org, 2008.

[BK02] F. Bause and P.S. Kritzinger. *Stochastic Petri Nets.* Vieweg, 2002.

[CCRFS06] L. Calzone, N. Chabrier-Rivier, F. Fages, and S. Soliman. Machine learning biochemical networks from temporal logic properties. *Trans. on Computat. Syst. Biol. VI, LNBI 4220*, pages 68–94, 2006.

[CF03] N. Chabrier and F. Fages. Symbolic model checking of biochemical networks. In *Proc. CMSB 2003*, pages 149–162. LNCS 2602, Springer, 2003.

[CGP01] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model checking.* MIT Press 1999, third printing, 2001.

[Cha07] C. Chaouiya. Petri net modelling of biological networks. *Briefings in Bioinformatics*, 8(4):210–219, 2007.

[Cha08] Charlie Website. A Tool for the Analysis of Place/Transition Nets. BTU Cottbus, http://www-dssz.informatik.tu-cottbus.de/software/charlie/charlie.html, 2008.

[CKS07] V. Chickarmane, B.N. Kholodenko, and H.M. Sauro. Oscillatory dynamics arising from competitive inhibition and multisite phosphorylation. *Journal of Theoretical Biology*, 244(1):68–76, January 2007.

[Cur06] E. Curry. Stochastic simulation of the entrained circadian rhythm. Master Thesis, School of Informatics, Univ. of Edinburgh, 2006.

[CVGO06] M. Calder, V. Vyshemirsky, D. Gilbert, and R. Orton. Analysis of signalling pathways using continuous time Markov chains. *Trans. on Computat. Syst. Biol. VI, LNBI 4220*, pages 44–67, 2006.

[DA05] R. David and H. Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer, 2005.

[DDS04] D. D'Aprile, S. Donatelli, and J. Sproston. CSL model checking for the GreatSPN tool. In *Proc. ISCIS 2004, LNCS 3280*, pages 543–552, 2004.

[DE95] J. Desel and J. Esparza. *Free Choice Petri Nets*. Cambridge University Press, New York, NY, USA, 1995.

[DJ01] J. Desel and G. Juhás. What is a Petri net? In *Unifying Petri Nets - Advances in Petri Nets*, pages 1–25, Tokyo, 2001. LNCS 2128, Spinger.

[DP80] J. R. Dormand and P. J. Prince. A family of embedded Runge-Kutta formulae. *J. Comp. Appl. Math.*, 6:1–22, 1980.

[EKL+02] S. Eker, M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and K. Sonmez. Pathway logic: Symbolic analysis of biological signaling. In *Proc. Seventh Pacific Symposium on Biocomputing*, pages 400–412, 2002.

[FR07] F. Fages and A. Rizk. On the analysis of numerical data time series in temporal logic. In *Proc. CMSB 2007*, pages 48–63. LNCS/LNBI 4695, Springer, 2007.

[GH06] D. Gilbert and M. Heiner. From Petri nets to differential equations - an integrative approach for biochemical network analysis. In *Proc. ICATPN 2006*, pages 181–200. LNCS 4024, Springer, 2006.

[GHL07a] D. Gilbert, M. Heiner, and S. Lehrack. A unifying framework for modelling and analysing biochemical pathways using Petri nets. TR I-02, CS Dep., BTU Cottbus, 2007.

[GHL07b] D. Gilbert, M. Heiner, and S. Lehrack. A unifying framework for modelling and analysing biochemical pathways using Petri nets. In *Proc. CMSB 2007*, pages 200–216. LNCS/LNBI 4695, Springer, 2007.

[GHR+08] D. Gilbert, M. Heiner, S. Rosser, R. Fulton, Xu Gu, and M. Trybiło. A case study in model-driven synthetic biology. In *2nd IFIP Conference on Biologically Inspired Collaborative Computing (BICC), IFIP WCC 2008, Milano*, to appear, 2008.

[Gil77] D.T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[GP98] P.J.E. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. *Proc. Natl. Acad. Sci. USA*, 95(June):2340–2361, 1998.

[HDG08] M. Heiner, R. Donaldson, and D. Gilbert. *Petri Nets for Systems Biology, in Iyengar, M.S. (ed.), Symbolic Systems Biology: Theory and Methods*. Jones and Bartlett Publishers, Inc., to appear, 2008.

[HFS+03] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, and et al. The Systems Biology Markup Language (SBML): A medium for

representation and exchange of biochemical network models. *J. Bioinformatics*, 19:524–531, 2003.

[HK04] M. Heiner and I. Koch. Petri net based model validation in systems biology. In *Proc. 25th ICATPN 2004, LNCS 3099*, pages 216–237. Springer, 2004.

[HKN+06] J. Heath, M. Kwiatkowska, G. Norman, D. Parker, and O. Tymchyshyn. Probabilistic model checking of complex biological pathways. In *Proc. CMSB 2006*, pages 32–47. LNBI 4210, Springer, 2006.

[Hof94] R. Hofestädt. A Petri net application of metabolic processes. *Journal of System Analysis, Modeling and Simulation*, 16:113–122, 1994.

[HRS08] M. Heiner, R. Richter, and M. Schwarick. Snoopy - a tool to design and animate/simulate graph-based formalisms. In *Proc. PNTAP 2008, associated to SIMUTools 2008*. ACM digital library, 2008.

[KH08] I. Koch and M. Heiner. *Petri Nets, in Junker B.H. and Schreiber, F. (eds.), Biological Network Analysis*, chapter 7, pages 139 – 179. Wiley Book Series on Bioinformatics, 2008.

[Kie02] A. M. Kierzek. STOCKS: STOChastic kinetic simulations of biochemical systems with Gillespie algorithm. *Bioinformatics*, 18(3):470–481, 2002.

[Lau73] K Lautenbach. Exact Liveness Conditions of a Petri Net Class (in German). Technical report, GMD Report 82, Bonn, 1973.

[LBS00] A. Levchenko, J. Bruck, and P.W. Sternberg. Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties. *Proc Natl Acad Sci USA*, 97(11):5818–5823, 2000.

[MBC+95] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing, John Wiley and Sons, 1995. 2nd Edition.

[MC208] MC2 Website. MC2 - PLTL model checker. University of Glasgow, http://www.brc.dcs.gla.ac.uk/software/mc2/, 2008.

[MFD+03] H. Matsuno, S. Fujita, A. Doi, M. Nagasaki, and S. Miyano. Towards pathway modelling and simulation. In *Proc. 24th ICATPN, LNCS 2679*, pages 3–22, 2003.

[MGF] Max-Gruenebaum-Foundation. http://www.max-gruenebaum-stiftung.de.

[MLM06] H. Matsuno, C. Li, and S. Miyano. Petri net based descriptions for systematic understanding of biological pathways. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, E89-A(11):3166–3174, 2006.

[MSS05] W. Marwan, A. Sujathab, and C. Starostzik. Reconstructing the regulatory network controlling commitment and sporulation in *Physarum polycephalum* based on hierarchical Petri net modeling and simulation. *J. of Theoretical Biology*, 236(4):349–365, 2005.

[Mur89] T. Murata. Petri nets: Properties, analysis and applications. *Proc.of the IEEE 77*, 4:541–580, 1989.

[Pal06] B.O. Palsson. *Systems Biology: Properties of Reconstructed Networks*. Cambridge University Press, 2006.

[Pet62] C.A. Petri. *Communication with Automata (in German)*. Schriften des Instituts für Instrumentelle Mathematik, Bonn, 1962.

[PNK06] D. Parker, G. Norman, and M. Kwiatkowska. *PRISM 3.0.beta1 Users' Guide*, 2006.

[Pnu81] A. Pnueli. The temporal semantics of concurrent programs. *Theor. Comput. Sci.*, 13:45–60, 1981.

[PW03] L. Priese and H. Wimmel. *Theoretical Informatics - Petri nets (in German)*. Springer, 2003.

[Rei82] W. Reisig. *Petri nets; An introduction.* Springer, 1982.

[RML93] V.N. Reddy, M.L. Mavrovouniotis, and M.L. Liebman. Petri net representations in metabolic pathways. In *'Proc. of the Int. Conf. on Intelligent Systems for Molecular Biology'*, 1993.

[SK05] H. Salis and Y. Kaznessis. Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *J. Chem. Phys.*, 122(054103), 2005.

[Sno08] Snoopy Website. A Tool to Design and Animate/Simulate Graphs. BTU Cottbus, http://www-dssz.informatik.tu-cottbus.de/software/snoopy.html, 2008.

[SPB01] R. Srivastava, M.S. Peterson, and W.E. Bentley. Stochastic kinetic analysis of the *escherichia coli* stress circuit using σ^{32}-targeted antisense. *Biotechnology and Bioengineering*, 75(1):120–129, 2001.

[SR97] L.F. Shampine and M.W. Reichelt. The MATLAB ODE Suite. *SIAM Journal on Scientific Computing*, 18:1–22, 1997.

[SR99] P.H. Starke and S. Roch. *INA - The Intergrated Net Analyzer.* Humboldt University Berlin, www.informatik.hu-berlin.de/∼starke/ina.html, 1999.

[SSE03] C. Schröter, S. Schwoon, and J. Esparza. The Model Checking Kit. In *Proc. ICATPN, LNCS 2679*, pages 463–472. Springer, 2003.

[SSW05] O.J. Shaw, L.J. Steggles, and A. Wipat. Automatic parameterisation of stochastic Petri net models of biological networks. CS-TR-909, School of CS, Univ. of Newcastle upon Tyne, 2005.

[ST05] O. Schulz-Trieglaff. Modelling the randomness in biological systems. Master Thesis, School of Informatics, University of Edinburgh, 2005.

[Sta89] P.H. Starke. Some properties of timed nets under the earliest firing rule. *Advances in Petri Nets*, LNCS 424:418–432, 1989.

[Sta90] P.H. Starke. *Analysis of Petri Net Models (in German).* B. G. Teubner, Stuttgart, Stuttgart, 1990.

[Ste94] W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton Univ. Press, 1994.

[Tov06] Tovchigrechko, A. *Model checking using interval decision diagrams.* PhD thesis, BTU Cottbus, Dep. of CS, submitted 2006.

[TSB04] T.E. Turner, S. Schnell, and K. Burrage. Stochastic approaches for modelling in vivo reactions. *Comp. Biology and Chemistry*, 28(3):165–178, 2004.

[Wil06] D.J. Wilkinson. *Stochastic Modelling for System Biology.* CRC Press, New York, 1st Edition, 2006.

[YKNP06] H. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *STTT*, 8(3):216–228, 2006.

[YS02] H.L.S. Younes and R.G. Simmons. Probabilistic verification of descrete event systems using acceptance sampling. In *Computer Aided Verification*, pages 223–235. LNCS 2404, Springer, 2002.

Remark This paper is a substantially extended version of [GHL07b]. The complete model specifications are given in the appendix of [GHL07a]. The data files of the running example in its three versions and the analysis results are available at www-dssz.informatik.tu-cottbus.de/examples/levchenko.